LAUREA
AMMATTIKORKEAKOULU
Uuden edellä

# Sanomaliikenteen tilastointisovelluksen kehittäminen Caché Objectscript -ohjelmointikielellä

Torkkeli, Juho

2014 Kerava

**Laurea-ammattikorkeakoulu**
Kerava

# Sanomaliikenteen tilastointisovelluksen kehittäminen Caché Objectscript -ohjelmointikielellä

Torkkeli, Juho
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Lokakuu, 2014

**Laurea-ammattikorkeakoulu** **Tiivistelmä**
Laurea Kerava
Tietojenkäsittelyn koulutusohjelma

Torkkeli, Juho

**Sanomaliikenteen tilastointisovelluksen kehittäminen Caché Objectscript -ohjelmointikielellä**

Vuosi 2014 Sivumäärä 51

Tämän opinnäytetyön tavoitteena oli kehittää Tietotarha Oy:lle Caché Objectscript -ohjelmointikieltä käyttäen sovellus, jonka avulla voidaan lukea Biztalk-palvelimen tuottamia sanomaliikenteen virhelokitiedostoja. Virhelokitiedostojen sisältämät sanomien virheelliset tapahtumat luetaan sovelluksen avulla Caché-tietokantaan, josta ne avataan HTML-taulukkona selaimeen. Taulukosta voidaan näppärästi etsiä ja havaita haluttuja asioita suuresta tietomäärästä.

Työssä käydään lävitse Caché Objectscript –ohjelmointikieltä ja sovelluksen kehittämistä. Työssä selitetään lyhyesti, miten projektiin liittyy Microsoft Biztalk, Caché & Caché Zen, HL7 sekä MUMPS. Työssä esitellään yleisempiä virhetilanteita, jotka voivat muodostua sanomalle sanomaliikenteessä prosessin aikana.

Työ sisältää projektikuvauksen sovelluksen kehityskulusta. Aineistoa kerättiin laajasta InterSystemsin dokumentaatiosta, kirjallisuudesta ja haastattelun avulla. Tutkimusmenetelmänä käytetään kvalitatiivista tutkimusmenetelmää.

Työn pääasiallisena tarkoituksena, sovelluksen kehittämisen lisäksi, oli tutustua Caché Objectscript –ohjelmointikieleen ja kouluttaa yritykselle Caché-osaaja. Ohjelmointikieltä lähdettiin tutkimaan vuoden 2013 kesällä ja sovellusta kehittämään saman vuoden syksynä. Sovellusta tullaan jatkokehittämään ja tarkoituksena on tehdä siitä täysin automaattinen työkalu case-yrityksen henkilökunnan käyttöön tulevaisuudessa.

Asiasanat: Caché-tietokanta, Caché Objectscript, sanomaliikenne, sovelluskehitys

**Laurea University of Applied Sciences** **Abstract**
Laurea Kerava
Bachelor's Degree Programme in Business Information Technology

Torkkeli, Juho

**Statistical Application Development for Message Traffic with Caché ObjectScript Programming Language**

Year 2014 Pages 51

The aim of this Bachelor's thesis was to develop an application for Tietotarha Oy using Caché ObjectScript programming language, which reads erroneous traffic event logs created by BizTalk server. The data contained within erroneous event logs is saved with the application to Caché database, and from there opened with a HTML table to a browser. The table can be used to search and detect various things from a massive amount of information.

This thesis studies Caché ObjectScript programming language and application development. Thesis briefly explains how the project relates to Microsoft BizTalk, Caché & Caché Zen, HL7 and MUMPS. The thesis introduces the most common traffic error conditions which may consist to messages during the process.

The thesis includes a project description of the application's development. The data was gathered from InterSystems Caché documentation, literature and interviewing. A qualitive research method was used in this thesis.

The main focus of this thesis, along with the application development, was to introduce me to Caché ObjectScript programming language and to train a Caché expert for the company. The examining of the programming language started in the summer of 2013 and application development on autumn. This same application will continue to be developed and the intent is to create a fully automatic tool for the rest of the staff of the case company in the future.

Keywords: Caché Database, Caché ObjectScript, Message Traffic, Application Development

Sisällys


1 Johdanto ..... 6
1.1 Projektin tavoite ..... 7
1.2 Projektin vaiheiden kuvaus ..... 8
1.3 Tutkimusmenetelmät ..... 9
1.4 Tutkimuskysymykset ..... 9
1.5 Reliabiliteetti ja validiteetti ..... 10
2 Caché moniulotteisen tiedon käsittelijänä ..... 10
2.1 Cachén edeltäjä MUMPS ..... 11
2.2 Cachén vahvuudet ..... 11
2.2.1 Merkkijonojen käsittelykyky ..... 12
2.2.2 Tehokas tilankäyttö ..... 13
2.2.3 Alaindeksien merkintä taulukoissa ..... 14
2.2.4 Useiden prosessien yhtäaikainen suoritus ..... 14
2.3 Kritiikkiä Cachésta ..... 15
2.4 Caché Zen ..... 16
3 Biztalk integraatiopalvelin ..... 17
3.1 Sanoman kulku ..... 17
3.2 Sanoman rakenne HL7-standardin pohjalta ..... 19
3.3 Virhetietojen muodostus tapahtumalokista ..... 20
4 Sovelluksen esittely ..... 21
4.1 Sovelluksen määrittely ..... 22
4.2 Sovelluksen toteutus ..... 22
4.2.1 Tiedostojen nimien selvittäminen hakemistopolusta ..... 23
4.2.2 Tiedostojen avaus ..... 24
4.2.3 Rivien luku ja muokkaus ..... 25
4.2.4 Olioiden luonti ja tallentaminen tietokantaan ..... 26
4.2.5 Tiedostojen siirto ..... 26
4.3 Sovelluksen testaus ..... 27
4.4 Selaimessa toimivan taulukon luonti Caché Zenillä ..... 29
4.5 Caché-tietokannan tietojen avautuminen selaimeen ..... 30
4.6 Sovelluksen ensimmäisen vaiheen lopputulos ..... 31
4.7 Sovelluksen jatkokehitys ..... 32
5 Johtopäätökset ..... 33
Kuvat ..... 37
Taulukot ..... 40
Liitteet ..... 41

# 1 Johdanto

Tämän opinnäytetyön tavoitteena on kehittää tilastointisovellus Tietotarha Oy:lle. Sovelluksen avulla luetaan Biztalk-palvelimen tuottamia virheellisiä tapahtumia sisältäviä virhelokitiedostoja. Virhelokitiedostot tallennetaan Caché-tietokantaan, josta ne avataan HTML-taulukkona selaimeen. Taulukosta voidaan hakea esimerkiksi tiettyjen sanomaliikenteessä tapahtuneiden virheiden lukumääriä.

Sanomaliikenteessä voi helposti syntyä vuorokauden aikana paljon virheellisiä tapahtumia. Esimerkiksi sanomalle vaadittavat tietokentät on voitu määritellä puutteellisesti ohjelmassa, sanoman lähettäjä ei saa kuittausta vastaanottajalta, tai yhteyttä ollenkaan tietoliikenneongelmien vuoksi. Kaikki sanomalle tapahtuvat asiat kootaan Biztalkin tapahtumalokiin (Event Log) ja lokijäsentimen (Log Parser) avulla niistä erotetaan erikseen sanomat, joille on tapahtunut jonkinlainen virhetilanne.

Projekti alkoi vuoden 2013 syyskuussa heti Caché Objectscript (COS) -ohjelmointikielen perusteiden opettelun jälkeen. Sain tehtäväksi yksinkertaisia tehtäviä aina varsinaiseen toimeksiantoon, joiden tarkoituksena oli tutustuttaa minut nollatason lähtötiedoilla uuteen ohjelmointikieleen, ja tuottaa ensimmäinen hyödyllinen sovellus muun henkilökunnan tarpeisiin.

Projektin esittelyn jälkeen työssä käsitellään Cachéta ja erityisemmin sen mukana tullutta COS–ohjelmointikieltä. Myös muita sovellukseen liittyviä asioita käsitellään lyhyesti, kuten Cachén edeltäjää MUMPSia ja Caché Zeniä. Työssä selvitetään Biztalk-palvelimen arkkitehtuuria ja sanoman rakennetta HL7-standardin pohjalta, sekä mistä sanomista ja niiden virhetilanteista on kysymys. Sovelluksen esittely -kappaleessa selitetään tarkemmin kehitettyä sovelluksen kokonaisuutta ja esitetään tehtyä koodia yksityiskohtaisemmin.

Opinnäytetyö toteutettiin kvalitatiivisena eli laadullisena kehittämishankkeena ja sain vapaat kädet sen kehittämisen aikana. Muutamien sovellusta käsittelevien projektipalaverien aikana sain suuntaa antavia neuvoja ja lisätoiveita sovelluksen kehittämisen kannalta. Tässä projektissa keskitytään pääasiallisesti Cachén mukana tulleeseen COS-ohjelmointikieleen ja itse sovelluksen kehittämiseen sillä ja muilla Cachén työkaluilla.

## 1.1 Projektin tavoite

Lähdin työstämään Tietotarhan toimitusjohtajan kanssa aihetta opinnäytetyölleni. Sovimme että alan opiskelemaan yrityksen tulevaisuutta silmällä pitäen COS-ohjelmointikieltä, ja työstämään sanomaliikenteen tilastointisovellusta, jonka samalla dokumentoin opinnäytetyötä varten. Ennen sovelluksen teon varsinaista aloittamista sain harjoitusta erään toisen työntekijän alaisuudessa, joka opetti minulle perusteet kyseisestä ohjelmointikielestä, ja näin saattoi minut alkuun kielen oppimisessa. Vaativan alun jälkeen olen pääasiassa itsenäisesti opiskellen tuottanut lähes kaiken tekemästäni sovelluksesta, ja vain joskus turvautunut yksittäisissä asioissa muiden Tietotarhan työntekijöiden puoleen.

Tämän sovelluksen teon aikana minun piti ottaa selville, mitä kaikkea tulen tarvitsemaan, mistä aloitan ja mihin saakka projektia dokumentoin opinnäytetyöhön. Lisäksi jouduin rajaamaan mitä tietoja ei näytetä opinnäytetyössä, kuten IP-osoitteita tai organisaatioiden nimiä. Projektin voidaan katsoa alkaneen siitä, kun sain käsiini ensimmäisen kerran virhelokiraportin (Liite 2) ja sen olevan valmis, kun sovellus toimii ongelmitta, reagoi yleisimpiin käyttäjän tekemiin virheisiin, tai virhelokitiedoston puutteellisuuteen. Lisäksi sovelluksen pitää pystyä tallentamaan Caché-tietokantaan virhelokitiedostosta virheelliset sanomatapahtumat, ja että niistä voidaan muodostaa vaivattomasti erikseen selaimessa toimiva taulukko.

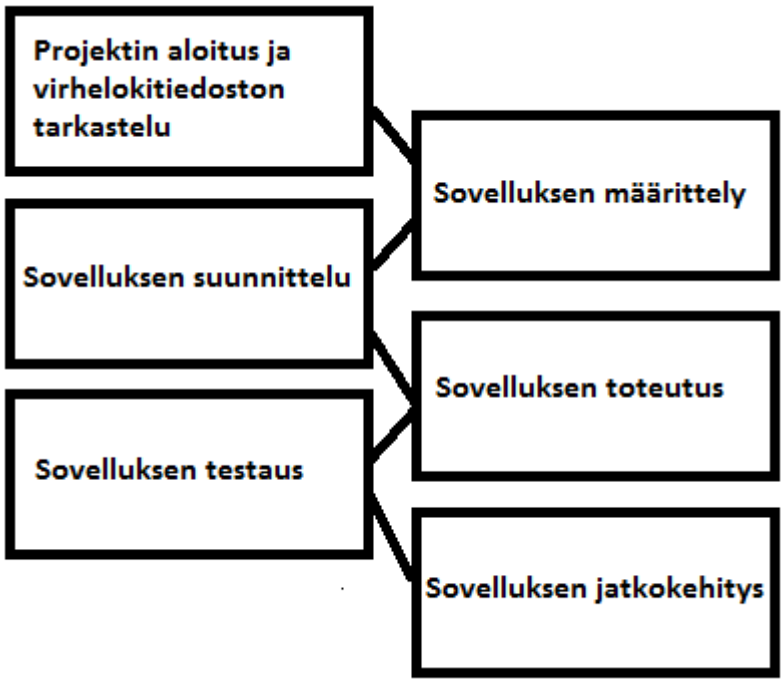
Suurin osa projektin tutkimustyöstä kului uusien kielien opettelemisessa ja englanninkielisen lähdeaineiston suomentamisessa. Joidenkin termien kohdalla jätin englanninkielisen vastineen sulkeisiin sanan perään, jos sen kääntäminen osoittautui hankalaksi tai käytetty termi oli tunnetumpi kuin suomenkielen vastaava. COS-ohjelmointikielen opettelu vaati oman aikansa aina sen perusteiden hallinnasta itsenäiseen tiedonkeruuseen. Muut Cachén ominaisuuksista, kuten Caché Zenin käytön oppiminen, vei oman aikansa. Aikataulu oli sovelluksen kehittämisen kannalta löysähkö, koska se riippui täysin siitä miten nopeasti opin tarvittavat asiat Cachésta. Tietoa löytyi runsaasti Cachén tekijän InterSystemsin englanninkielisestä dokumentaatiosta verkosta ja kirjallisuudesta. Lisäksi sain tietoa Biztalkista ja sanomaliikenteestä haastattelemalla Tietotarhan toimitusjohtajaa.

Sanomaliikenteessä kaikki mahdolliset asiat pitäisi pystyä automatisoimaan. Tämän vuoksi tehty sovellus täytyy saada aikanaan automatisoitua. Tämä opinnäytetyö sisältää ensimmäisen version sovelluksesta, joka voidaan manuaalisesti ajaa Caché-terminaalista. Seuraavassa luvussa kuvataan projektin vaiheet.

## 1.2 Projektin vaiheiden kuvaus

Projekti alkoi syksyllä 2013, kun virhelokitiedostoa alettiin tarkastelemaan. Alla olevassa kuviossa on karkeasti kuvattu projektin vaiheet (Kuvio 1). Tarkastelun jälkeen projektia käsittelevässä ylläpitopalaverissa todettiin, että tilastointisovelluksen kehittäminen voisi alkaa. Tämä projektina toteutettu kehittämishanke aloitettiin, kun Caché Objectscript -ohjelmointikielen perusteiden harjoittelu oli saatu päätökseen. Projektin aikana piti ottaa selvää Cachésta ja sen mukana tuodusta Caché Objectscript –ohjelmointikielestä. Projektin toivotaan tuovan laajalti tietoa Cachésta ja sovellukseen liittyvistä muista asioista, kuten Caché Zenistä.

Virhelokin tarkastelemisen jälkeen sovellukselle tehtiin vaatimusmäärittely, joka on kuvattu sovelluksen esittely -kappaleessa. Vaatimusmäärittelyn jälkeen sovellusta alettiin suunnitella. Sen pitäisi olla lopullisessa versiossaan täysin automaattinen. Ensimmäisessä vaiheessa pyritään saamaan sovellus toimivaan kuntoon ja sen jälkeen katsotaan, miten sovellus saataisiin automaattiseksi. Sovellusta lähdettiin toteuttamaan Caché-studio sovelluskehittimessä. Sovelluksen toteutus kesti syyskuusta seuraavan vuoden tammikuuhun, jonka jälkeen valmista sovellusta alettiin testata. Testauksen läpäistyä sille asetetut tavoitteet, alettiin seuraavissa projektia käsittelevissä palavereissa käsittelemään sen jatkokehitystä. Projektilla ei ollut ns. määriteltyä aikataulua, mutta se haluttiin kuitenkin valmiiksi mahdollisimman nopeasti.




Kuvio 1. Projektin eri vaiheet

## 1.3 Tutkimusmenetelmät

Tässä sovelluksen kehittämishankkeessa käytetään kvalitatiivista tutkimusmenetelmää. Työ perustuu itseymmärrykseen ja on luonteeltaan eksploratorista eli ”uusia ratkaisuja etsivää tutkivaa toimintaa” (Anttila 2008, 5). Työssä tarkastellaan Caché Objectscript -ohjelmointikieltä ja pyritään sen avulla tekemään toimiva sovellus. Työssä tullaan painottamaan sovelluksen teon yhteydessä COS-ohjelmointikielen ymmärtämistä. Ohjelmoinnissa asiat tehdään parhaaksi katsotulla tavalla ja pidetään huolta siitä, että tehty koodi vastaa hyvää ohjelmointitapaa, jota pyritään noudattamaan koko projektin ajan. Suurin osa sovelluksen tarvitsemasta koodista tehdään käytännön kokemuksen kautta.

Aineiston kerääminen COS-ohjelmointikielestä tutkitaan pääasiassa Cachén tekijän InterSystemsin dokumentaatiosta ja Caché ObjectScript and MUMPS -kirjasta. Sanomaliikenteestä tietoa hankitaan pääasiallisesti haastattelemalla Tietotarhan toimitusjohtajaa ylläpitopalaverien aikana.

Projektin kannalta jouduin itsenäisesti kartoittamaan Cachéta ja sen mukana tulleita asioita laajasta InterSystemsin dokumentaatiosta. Yrityksellä on ennestään osaamista MUMPS -ohjelmointikielestä, mutta ei riittävää tietotaitoa uudemmasta COS-ohjelmointikielestä. Minulle annettiin tehtäväksi sovelluksen kehittämisen lisäksi oppia tarpeeksi asioita Cachésta ja COS-ohjelmointikielestä, jotta opittua tietotaitoa voitaisiin mahdollisesti käyttää hyväksi tulevaisuudessa.

Hirsjärven, Remeksen ja Sajavaaran (2010, 138) mukaan kartoittava tutkimus etsii uusia näkökulmia tai ilmiöitä ja selvittää tunnettuja ilmiöitä. Työssä kartoitetaan COS -ohjelmointikieltä, sanoman kulkua Biztalk-palvelimen sisällä ja sanoman rakennetta HL7 -standardin pohjalta. Muutamien sovellusta käsittelevien yrityksen palaverien aikana sain ohjeita ja toiveita asioista, joita sovelluksen pitäisi pystyä tekemään.

## 1.4 Tutkimuskysymykset

Tämän projektin tarkoituksena oli selvittää miten Cachéta hyödyntäen ja COS -ohjelmointikielellä tehty sovellus toteutetaan käytännössä. Mitä etuja COS-ohjelmointikieli tuo mukanaan ja mistä sitä kritisoidaan. Miten virheelliset sanomatiedot avautuvat Cachélla tehtyyn taulukkoon ja minkälaisia ominaisuuksia siihen kuuluu. Biztalkin tapauksessa sen tarjoama tapahtumalokin käyttöliittymä sanomatilanteiden tarkastelulle ei ole tarpeeksi riittävä, joten on ensisijaisen tärkeää, että sanomien tarkastelu onnistuu helposti tehdyllä taulukolla. Toissijaisena tutkimuskysymyksenä selvitetään minkälaisia virheellisiä tilanteita virhelokista löytyy.

### 1.5 Reliabiliteetti ja validiteetti

Reliabiliteetilla tarkoitetaan mittaustulosten kykyä tuottaa vaadittuja tuloksia. Vaadittujen tulosten pitäisi olla ei-sattumanvaraisia mittausten toistuessa. Validiteetilla tarkoitetaan tutkimuksen luotettavuutta. Validiteetti voidaan arvioida esimerkiksi vertaamalla saatuja mittaustuloksia aikaisempiin havaintoihin. (Virtuaaliammattikorkeakoulu 2007a; Virtuaaliammattikorkeakoulu 2007b.)

Työssä selvitetään toissijaisena asiana virhelokien virheellisiä tapahtumia ja niiden lukumääriä. Selvitettäessä Caché-tietokannasta HTML-taulukkoon saatuja tuloksia, pitää taulukon osata järjestellä saatuja tietoja luotettavalla tavalla. Tässä tapauksessa pyritään varmentamaan, että työn aikana tehty HTML-taulukko osaa järjestellä tietoja tapahtuma-ID:n (Event ID) mukaan, jonka perusteella samantyyppisiä virhetilanteita voidaan verrata aikaisempiin havaintoihin sanomaliikenteessä.

## 2 Caché moniulotteisen tiedon käsittelijänä

Caché julkaistiin vuonna 1997 InterSystemsin kehittämänä korkean suorituskyvyn omaavana oliotietokantana (High-Performance Object DataBase), ja se koostuu useista Cachén liittyvistä komponenteista. Caché-tietokanta tarjoaa tiedon tallennuksen lomassa mahdollisuuden hallita useita prosesseja ja tapahtumia samanaikaisesti. Cachélla voidaan rakentaa monimutkaisia tietokantojen hallintajärjestelmiä ja sitä voidaan pitää tietokannan työkalupakkina. Caché on myös relaatiotietokanta, jonka avulla tietotauluja voidaan yhdistellä tietokannan sisällä. Tietotauluja pystytään hyödyntämään perinteisellä SQL (Structured Query Language) eli standardoidulla kyselykielellä ja muilla sitä tukevilla työkaluilla. Cachéta pidetään nopeana, luotettavana ja skaalautuvana relaatiotietokantana. (InterSystems 2013l.)

Caché tarjoaa laajan valikoiman erilaisia työkaluja. Muun muassa Caché Objectscript, Caché MVBasic ja Caché Basic -ohjelmointikielet, objektipohjaisen verkkopalvelutuen XML -merkintäkielellä (Extensible Markup Language), tekniikoita ja tarvikkeita jotka tarjoavat nopean kehityspohjan tietokanta- ja web-sovelluksille, automaattisen yhteensopivuuden useille muille ohjelmointikielille kuten ActiveX, C++, Java, JDBC, .NET, ODBC, Perl, Python, SOAP, XML. (InterSystems 2013l.)

Cachéta käytetään maailmanlaajuisesti monissa sovelluksissa ja järjestelmissä. Muutamina esimerkkeinä voidaan pitää suurta terveydenhuollon järjestelmää, johon kuuluu sovelluspalvelimien lisäksi 30000 työasemaa, verkossa toimivaa koulutusjärjestelmää tai

maailmanmestaruuskisojen aikana urheilijoiden reaaliaikaista nopeuden ja paikan mittausjärjestelmää. InterSystemsin mukaan Caché on valittu näihin hankkeisiin pääasiassa sen suorityskyvyn vuoksi. Sillä on myös nopeata kehittää järjestelmiä, joita ei kuitenkaan tarvitse alkaa kokonaan muokkaamaan uudelleen. (InterSystems 2013l.)

## 2.1 Cachén edeltäjä MUMPS

MUMPS eli Massachusetts General Hospital Utility Multi-Programming System (myöhemmin uudelleen nimetty Multi-User Multi-Programming System tai lyhennettynä M) on kehitetty 1960-luvulla terveydenhuoltoalan tarpeisiin. MUMPS on nopea ja tehokas tietokantajärjestelmä. Sen valttina pidetään sisäänrakennettua tietokantaa, joka mahdollistaa korkean tason yhteydet levylle käyttämällä yksinkertaisia symbolisia ohjelman muuttujia ja järjestettyä taulukointia (Subscripted Arrays). MUMPSin erikoisin piirre liittyy sen tietokantaan otetun yhteyden perusteella, joka tapahtuu muuttujien avulla eikä niinkään hauilla tai kyselyillä. Väliaikaiset ja pysyvät muuttujat käyttävät samaa perus syntaksia (säännöstö jonka avulla ilmaistaan ohjelmointikielen asettelu) tiedon varastoinnissa. Tämä takaa korkean suorituskyvyn tietojen saatavuuden kannalta. (Internet FAQ Archives 1999a; Kadow 2012, 1.)

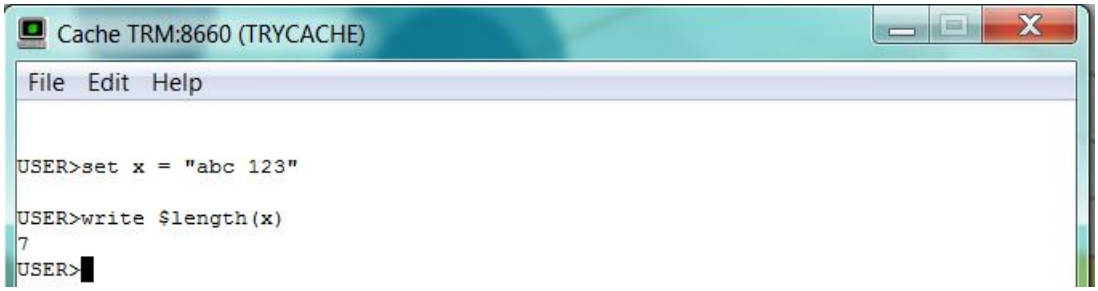
MUMPS-tietokannan taulukot käyttävät hyväksi B-tree tietorakennetta, jonka avulla tieto pysyy järjestettynä, mahdollistaa tiedon haun, MUMPSin tapauksessa peräkkäisen pääsyn ennalta määriteltyihin alaindekseihin (Subscript) sekä tiedon lisäämisen ja poistamisen siten, että käsittelyaika kasvaa logaritmisesti tietomäärään nähden. B-tree tietorakenne on optimaalinen tietokannoille, jotka käsittelevät suuria määriä tietoa. (Internet FAQ Archives 1999b.)

## 2.2 Cachén vahvuudet

Useiden muiden ohjelmointikielien lomassa Caché-ohjelmointi perustuu muuttujiin tallennettavan tiedon hallitsemiseen (Kadow 2012, 2). Caché antaa tätä varten käyttäjälle paljon erilaisia valmiiksi luotuja komentoja tai käskyjä (Command) ja toimintoja (Function) käytettäväksi järjestelmän puolelta, ja myös mahdollisuuden rakentaa itse omat toimintonsa. Käyttäjän puolelta rakennettuja toimintoja kutsutaan rutiineiksi (Routine) ja ne ovat Cachén mukana tulleiden ohjelmointikielien syntaksin mukaan tehtyjä ohjelmia. Rutiinien luomia muuttujia voidaan käyttää saman istunnon aikana muiden rutiinien kanssa. Tämä tarkoittaa sitä, että toisen rutiinin muuttujat ovat vapaasti käytettävissä, muutettavissa tai poistettavissa. Caché-ohjelmoinnissa käyttäjälle annetaan paljon vapauksia koodin kannalta, mutta vaaditaan kurinalaisuutta jotta koodi pysyy selkeänä. Seuraavissa alaluvuissa käsitellään Cachén vahvuuksia tarkemmin COS-syntaksin kannalta. (Kadow 2012, 29,31.)

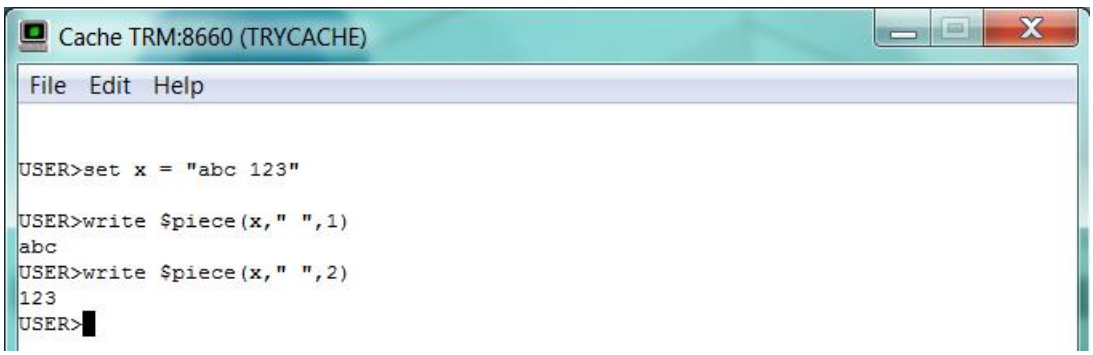
### 2.2.1 Merkkijonojen käsittelykyky

Yksi Cachén vahvuuksista on sen erinomainen merkkijonojen käsittelykyky. Merkkijono (String) voidaan muokata järjestelmän useilla eri toiminnoilla halutunlaiseksi. Ensimmäisenä esimerkkinä $Length –toiminto laskee sille annetun merkkijonon pituuden ja palauttaa takaisin numeerisen arvon (Kuva 1). Alla kuvattu komento ”set” laittaa muuttujaan ”x” arvon ”abc 123”. Toisella komennolla ”write” kirjoitetaan ulos toiminnolla ”$length” parametrin ”x” arvo. Arvoksi saadaan 7, koska myös tyhjät välit lasketaan mukaan merkkijonosta. (InterSystems 2013d; Kadow 2012, 31-32.)




Kuva 1. Toiminnon $Length -kuvaus

Toisena esimerkkinä voidaan pitää $Piece –toimintoa, jonka avulla merkkijonosta voidaan palauttaa käyttäjän haluama alimerkkijono (Substring) (Kuva 2). Tällä toiminnolla palautettu arvo muodostuu käyttäjän syöttämistä parametreista ”$piece” sulkujen sisään. Ensimmäisenä sulkujen sisällä olevasta parametrista ”x” annetaan seuraava arvo pilkulla eroteltuna, jota kutsutaan erottimeksi (Delimiter). Erottimella tarkoitetaan merkkiä, joka ei yksistään esiinny tekstissä, vaan sillä erotetaan muut halutut merkit toisistaan (Kadow 2012, 32). Erottimena toimii alla olevassa kuvassa välilyönti eli ” ”. Viimeinen arvo eli ”1” ja ”2” määrää mikä osa halutaan palauttaa takaisin erottimeen nähden. Ensimmäinen osa ennen välilyöntiä palauttaa arvon ”abc” ja toinen välilyönnin jälkeen arvon ”123”. Jos erottimia olisi enemmän kuin yksi voitaisiin palauttaa erottimeen nähden eri arvo. Esimerkiksi merkkijono ”abc 123 456” palauttaisi arvolla 3 merkkijonon ”456” jne. (InterSystems 2013i; Kadow 2012, 33.)




Kuva 2. Toiminnon $Piece -kuvaus

Yllä mainittujen esimerkkien lomassa Caché antaa monia muita merkkijonojen käsittelyn kannalta hyödyllisiä toimintoja. Aiheen laajuuden vuoksi muihin komentoihin ja toimintoihin kannattaa tutustua enemmän InterSystemsin sivuilta löytyvästä Caché-dokumentaatiosta.

### 2.2.2 Tehokas tilankäyttö

Kaikki tieto Caché-tietokannassa on sidottu moniulotteisiin oletusarvoja hyödyntäviin taulukoihin (Sparse Multidimensional Arrays) eli globaaleihin. Samaa arvoa hyödyntävät, eli tyhjillä tai NULL-arvoilla muodostetut taulukot, eivät vaadi että kaikkien tietokenttien pitäisi sisältää tietoa. Tämän ansiosta taulukot vaativat yleensä vain noin puolet siitä tilasta, mitä muut relaatiotietokannat tarvitsevat. Moniulotteisten taulukoiden avulla Cachéta käyttävistä sovelluksista tulee paljon joustavampia tiedon varastoinnin kannalta. Esimerkiksi jotkin toisiinsa läheisesti liittyvät oliot (Object) voidaan liittää fyysisesti yhteen samanlaiseen ryhmään (Cluster) tietokannan sisällä niin, että pääsy itse tietoon on tehokasta. (InterSystems 2013j.)

Globaaleja voidaan tallentaa tietokantaan joko yksittäisarvona (Scalar) tai taulukkona (Array), ja niiden varastoimaa tietoa levylle pidetään pysyvänä. Globaalien vastakohtana pidetään paikallisia (Local) muuttujia, jotka eivät tallennu levylle ja poistuvat koneen muistista sen sulkeutuessa. Alla olevasta taulukosta nähdään miten paikallinen ja globaali -muuttuja eroavat ja esitetään toisistaan (Taulukko 1). Tiedon tallentuessa levylle muuttujan eteen laitetaan järjestelmäkohdistin (Caret) eli ”^”. Paikallisen muuttujan tapauksessa ei koskaan laiteta järjestelmäkohdistinta sen eteen. Taulukon kohdalla muuttujia viitataan niiden alaindeksien kanssa, jotka tässä tapauksessa näkyvät tarkemmin itse numerona ”X(1)” ja merkkijonoina ”JOHN(”ADDR”,”CITY”). Merkkijonon tapauksessa taulukon alaindekseillä voidaan tarkentaa tietokentässä olevaa tietoa tai ilmaista sitä jo itse alaindeksin kohdalla. Alaindekseistä ”ADDR” ja ”CITY” voidaan päätellä niiden sisältävän tietoa jostain osoitteesta ja kaupungista. Yksittäisillä arvoilla sisältävillä paikallisilla muuttujilla ja globaaleilla ei voi koskaan olla merkittynä alaindeksejä. (Kadow 2012, 27-28.)

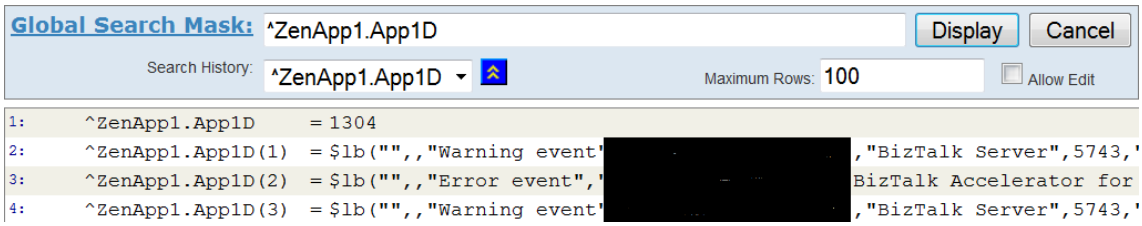
***Table 2-1: Local and Global Variable Examples***

| Variable Type | Local | Global |
|---|---|---|
| Scalar | X John | ^X ^John |
| Array | X(1)<br>JOHN("ADDR","CITY") | ^X(1)<br>^JOHN("ADDR","CITY") |

Taulukko 1. Esimerkki yksittäisarvona ja taulukkona esitetystä paikallisesta muuttujasta ja globaalista (InterSystems 2000.)

### 2.2.3 Alaindeksien merkintä taulukoissa

Yhtenä Cachén vahvuutena voidaan pitää sen alaindeksöintiä. Yllä olevassa luvussa huomataan, miten paikallisten ja globaalien -muuttujien käyttämät alaindeksit voidaan nimetä numeerisesti tai merkkijonona (Taulukko 1). Alaindeksiä ei voi kuitenkaan nimetä tyhjällä arvolla, vaikka itse tietokenttä voidaan esittää tyhjänä. Alaindeksit ottavat huomioon kirjaimen koon. Alla olevasta kuvasta, joka on otettu kehitetystä sovelluksesta, huomataan kyseessä olevan taulukko tallennetusta globaalista nimeltään ”^ZenApp1.App1D” ja sillä on kolme alaindeksiä numeroituna (Kuva 3). Itse tieto sisältyy kenttään yhtäsuuruus-merkin oikealle puolelle. Alaindeksin voisi nimetä ”^ZenApp1.App1D(”1!é@#$%^&*”) ja se olisi kelvollinen. (InterSystems 2013a.)




Kuva 3. Esimerkki taulukoidusta globaalista (organisaatioiden nimet on sensuroitu pois)

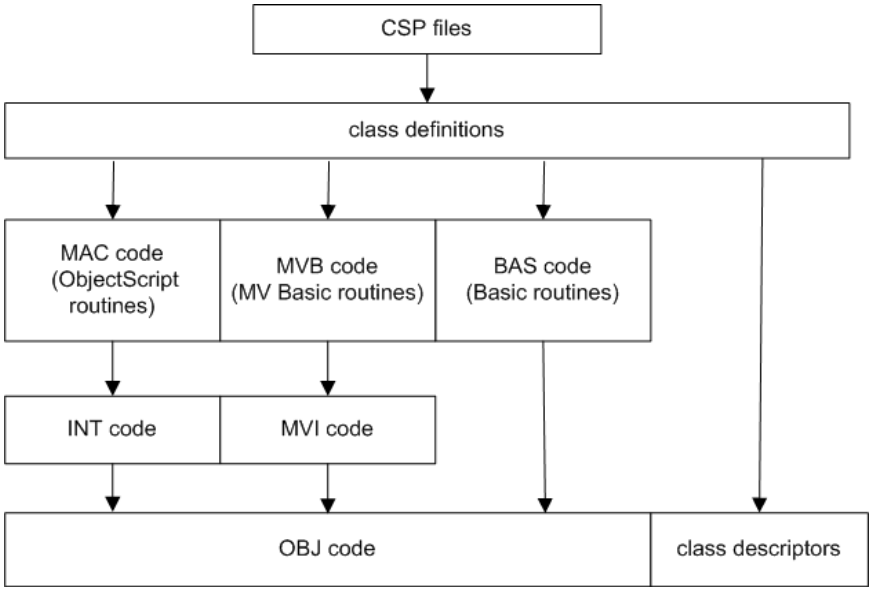
Alaindeksien merkintää numeroituna pidetään ihanteellisena, koska tietokenttien muokkaaminen yhtäsuuruus-merkin oikealta puolelta on helpompaa. Numeroituna globaalia ei tarvitse alkaa alustamaan uudelleen ja se pysyy vakaana. Näin tietotaulukko voidaan aina säätää tarpeen mukaan heijastamaan uutta liiketoimintamallia. (Kadow 2012, 104.)

### 2.2.4 Useiden prosessien yhtäaikainen suoritus

Cachélla voidaan suorittaa tuhansia tietokantaprosesseja samanaikaisesti. Prosessiksi kutsutaan taustalla ajettavaa virtuaalikoneen järjestelmäprosessia tai ohjelman yksittäistä ilmentymää, jota ollaan ajamassa Cachén palvelimella. Jokaisella prosessilla on tehokas ja suora yhteys tietokantaan. Virtuaalikone suorittaa ohjeita, joita kutsutaan P-koodiksi (P -code). P-koodi on optimoitu hyvin tietokantaa, I/O- (Input/Output) ja loogisia operaatioita varten, joita vaaditaan yleensä liiketoimintaprosesseilta ja tiedon varastointia tekeviltä sovelluksilta. Virtuaalikoneen koodia voidaan luoda SQL-kyselyiden avulla, jotka muutetaan P -koodiksi. Valmista P-koodia tuottaa myös Cachén omat metodigeneraattorit, joita käytettiin Caché Zenin kanssa luodessa sovellukselle taulukko (Liite 3). Cachén mukana tulleilla ohjelmointikielillä Caché Objectscriptillä ja Caché Basicillä tehdyt sovellukset voivat käyttää olio-metodeja, jotka voivat sisältää mitä tahansa logiikkaa käyttäjä haluaa. Työssä kuvattu sovellus tehdään Caché Objectscriptillä (Liite 1), josta näkyy että sen pääohjelman teko aloitetaan luokkametodilla ”ClassMethod lueTiedosto(saveFromLine = 0)”. Kun sovellusta lähdetään ajamaan palvelimelta, P-koodi tulkkaa sen sisältämät ohjelmointikielet. P-koodi on

välikoodina erikseen räätälöity yhteensopivaksi eri koneiden ja käyttöjärjestelmien kanssa. Näin Cachélla tehdyt ohjelmat lähtevät pyörimään riippumatta käytettävästä koneesta tai käyttöjärjestelmästä. (InterSystems 2013k.)

Cachén mukana tulleita monia eri ohjelmointikieliä ja elementtejä voidaan käyttää hyväksi samanaikaisesti rutiineissa tai määritellyissä luokissa. Käyttäjän kirjoittamaa koodia ei käytetä kuitenkaan suoraan sellaisenaan, vaan koodin kokoamisvaiheessa (Compile Time) Caché tuottaa kaikesta kirjoitetusta koodista OBJ-koodia, jonka virtuaalikone lopuksi tulkkaa P-koodin avulla. Alla olevan kaavion perusteella huomataan miten eri elementit muunnetaan pala palalta (Kaavio 1). CSP (Caché Server Page) –moottori muuntaa CSP-tiedostot määritellyiksi luokiksi (Class Definition). Luokkien kokoajat (Class Compiler) käyttävät määriteltyjä luokkia ja tuottavat MAC, MVB ja BAS –koodia riippuen käytettävistä ohjelmointikielistä. Luokan kokoaja tuottaa myös avainsanan (Class Descriptor) kullekin luokalle ja käyttää tätä ohjelman suorituksen aikana (Runtime). Caché Objectscriptin käyttämä MAC-koodi käännetään vielä erilliselle INT-välikoodille, ja samoin käy myös MV Basicin kanssa, joka käännetään MVI-välikoodille. Basic ei käytä erikseen ylimääräistä välikoodia. Tämän jälkeen eri välikoodit käännetään OBJ-koodiksi. Cachén virtuaalikone käyttää tätä OBJ-koodia ja kokoamisvaiheen jälkeen näitä välikoodeja ei enää tarvita. (InterSystems 2013c.)

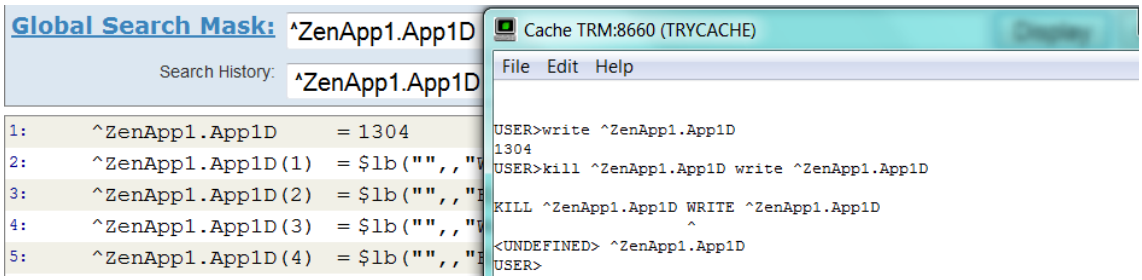



Kaavio 1. Cachén elementtien kokoamisvaiheet (InterSystems 2013c.)

## 2.3 Kritiikkiä Cachésta

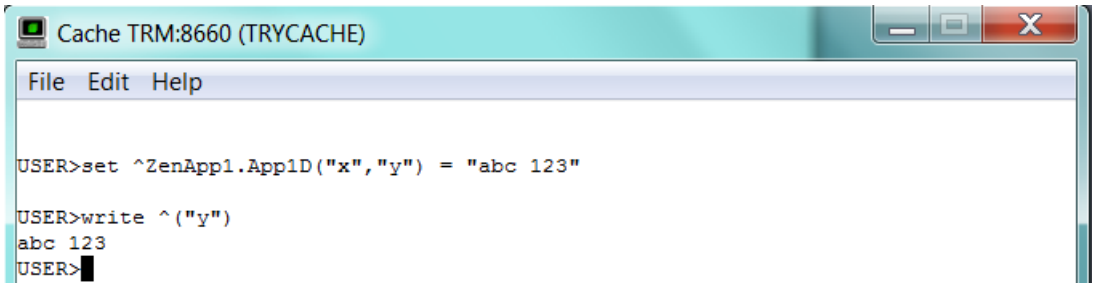
Caché ei ole kuitenkaan kovin käyttäjäystävällinen ja vaatii myös kokeneelta käyttäjältä tarkkaavaisuutta. Cachélla voidaan helposti ja nopeasti luoda koodia sen mukana tuomilla toiminnoilla, mutta ne saattavat tehdä ohjelmista kryptisiä. Cachén kannalta suurin kritiikki

kohdistuu sen erittäin tehokkaaseen Kill-komentoon, jonka avulla käyttäjä pystyy hetkessä pyyhkimään tietokannasta globaalin ilman, että sitä voidaan enää palauttaa. Toisaalta poistetut tiedot voidaan palauttaa varmuuskopiosta, jos se on määritelty erikseen. Alla olevan esimerkin mukaan globaali ”^ZenApp1.App1D” tuhotaan komennolla ”kill” (Kuva 4). Heti kun globaali on tuhottu ja se yritetään kirjoittaa komennolla ”write”, saadaan virheilmoitus ”<UNDEFINED>”, joka tarkoittaa ettei globaalia ole määritelty. (Kadow 2012, 114.)




Kuva 4. Esimerkki Kill-komennosta

Yksi Cachén nopeista ohjelmointitavoista on käyttää hyväksi paljaita viittauksia. Näitä ei kuitenkaan katsota ohjelmien ylläpitämisen kannalta hyväksi tavaksi. Paljailla viittauksilla tarkoitetaan Cachén muistissa olevan viimeisimmän globaalin käyttämistä. Alla olevan esimerkin mukaisesti paljaalla viittauksella oletetaan, että globaalin ”^ZenApp1.App1D” koko nimi on jo muistissa, ja käyttäjä on liian laiska kirjoittamaan sitä uudelleen (Kuva 5). Näin käyttäjä saa tiedon ulos globaalista, mutta jos esimerkiksi jokin toinen globaali olisi tehty ennen ”write” -komentoa, olisi viittaus siirtynyt uusimpaan globaaliin. Myös täsmennettäessä alemmalla tasolla olevaa saman globaalin alaindeksiä, siirtyy viittaus sille tasolle. Tällä tavalla voitaisiin helposti esimerkiksi Kill-komennon yhteydessä poistaa kokonaan väärä tietue tai tietueen osa. (Kadow 2012, 109.)




Kuva 5. Esimerkki paljaasta viittauksesta globaaliin

## 2.4 Caché Zen

Caché Zen antaa käyttäjälleen helpon tavan luoda nopeasti monimutkaisia verkkosovelluksia valmiiksi kootuilla olio-komponenteilla. Näillä komponenteilla luodaan automaattisesti HTML (Hypertext Markup Language) eli hypertekstin merkintäkieltä ja Javascriptiä, joita tarvitaan monimutkaisten web-sovelluksien rakentamiseen. Lisäksi ne tarjoavat yhteisen oliomallin

(Object Model), joka on yhteiskäytössä selaimen ja palvelimella ajettavan sovelluksen kanssa. (InterSystems 2013b.)

Caché Zen perustuu CSP (Caché Server Page) ja Caché oliotietokanta (Cache Object Database) –teknologioihin. Nämä teknologiat tarjoavat kestävän, skaalautuvan ja siirrettävän alustan web-sovelluksille. Zenillä web-sovellusten kehitys on helppoa CSP:n perusominaisuuksien kuten suorituskyvyn, tietojen saatavuuden, turvallisuuden, lokalisoinnin ja kokoonpanon avulla. (InterSystems 2013b.)

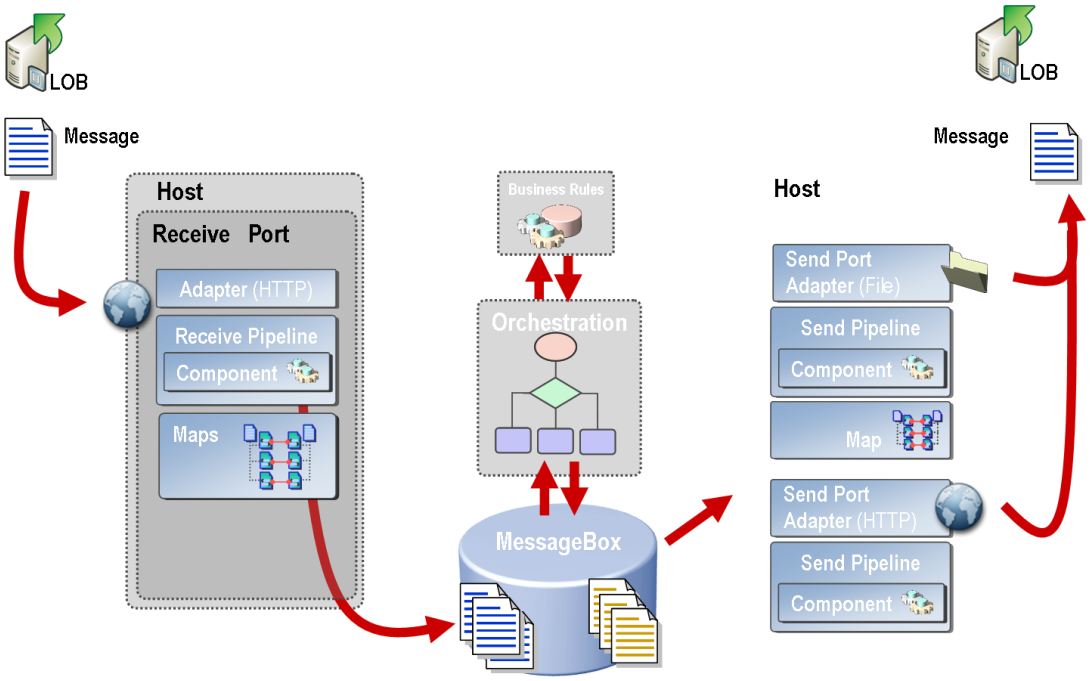
# 3 Biztalk integraatiopalvelin

Biztalk on vuonna 2000 alkunsa saanut Microsoftin kehittämä integraatio-ohjelmisto, jonka avulla voidaan automatisoida erilaiset ohjelmat ja sovellukset toimimaan tehokkaasti keskenään. Biztalk antaa käyttäjälle tätä varten kokoelman erilaisia työkaluja, muun muassa orkestraation (Orchestration Designer), XLANG-kielen aikatauluttajan (XLANG Scheduler), sanomamuodon editorin (Editor), sanomien kartoittajan (Mapper), sanomamanagerin (Messaging Manager), sanomapalvelut (Messaging services), asiakirjojen seurannan (Document Tracking) ja palvelinhallinnon (Server Administration). (Mohr & Woodgate 2000, 24-25; Tarhonen 2014b.)

Biztalkia käytetään Windows-käyttöjärjestelmän kanssa, mutta sitä ei ole sidottu vain Windowsia käyttävien ohjelmien integrointiin. Biztalk on tarkoitettu käytettäväksi eri järjestelmien yhdistämiseen sanomaliikenteen avulla, ja sen perusominaisuudet ovat pysyneet lähes muuttumattomina vuoden 2000-versiosta uusimpaan 2013-versioon. (Tarhonen 2014b.)

## 3.1 Sanoman kulku

Biztalkiin tullessa kaikki sanomatyypit muutetaan XML-muotoon. Jokaisen sanomatyypin kohdalla Biztalkin eri jäsentäjät (Parser) ottavat niille luoduista XML-skeemoista (dokumentin asettelun ohjeet) mallia, miten tietty sanomatyyppi muunnetaan oikeaan XML-muotoon. Tämä prosessi tapahtuu kaikille määritellyille sanomatyypeille sekä Biztalkiin sisääntultaessa ja sieltä ulosvietäessä. Toisaalta sanoma voidaan välittää Biztalkin kautta myös sellaisenaan ilman, että se muutetaan XML-muotoon. Näitä sanomia viedään Biztalkiin sisään passthru-pipelinen kautta, mutta ne ovat XML-kehyksen sisällä Biztalkin tietokannassa. Seuraavalla sivulla olevan Biztalk-palvelimen arkkitehtuurikuvauksen sain haastattelemalla Tietotarhan toimitusjohtajaa ylläpitopalaverin aikana (Kuva 6). Kyseisen kuvauksen termejä en lähtenyt kääntämään, koska englanninkielisiä vastineita käytetään yleisesti alan asiantuntijoiden keskuudessa. (Mohr & Woodgate 2000, 520,522; Tarhonen 2014b.)




Kuva 6. Biztalk-palvelimen arkkitehtuuri (Tarhonen 2014a, 42.)

Yllä olevan kuvan mukaisesti lähetetty sanoma (Message) lähetetään Biztalkiin LOBilta (Line Of Business), joka saattaa olla esimerkiksi jokin organisaatio A ja se halutaan muuttaa Biztalkissa sopivaan muotoon ennen, kuin se lähetetään B organisaatioon (Kuva 6). Biztalkissa receive locationit sijoittuvat loogisen receiveportin alle. Receive locationissa määritellään käytettävä adapteri. Jokaiselle protokollalle esimerkiksi HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol) tai MLLP (Minimum Lower Layer Protocol) on oma adapterinsa, joka osaa kyseisen protokollan. Receive locationissa määritellään myös receive pipeline, joka osaa kyseiseen tietovirtaan liittyvän standardin esimerkiksi HL7 (Health Level Seven) V.2.x, EDIFACT tai XML. Receive pipeline muuttaa sanoman Biztalkin sisäiseen XML-muotoon. Sanoma säilyy muuttumattomana sisällöltään jos käytetään passthru-pipelinea. Receive pipeline kirjoittaa sanoman messageboxiin. Receive pipelineen voidaan kytkeä omia lisäkomponentteja, kuten esimerkiksi sanoman korjauksen. Receive pipelinessa voidaan mappauksen avulla muuttaa sanomaa. (Tarhonen 2014b.)

Messageboxista sanoma tilataan joko send-porttiin tai orkestraatioon (integraatio-ohjelmaan). Orkestraatiossa sanomaa voidaan muuttaa monipuolisella XLANG-ohjelmointikielellä, mutta esimerkiksi c#-kutsut ovat mahdollisia. Orkestraatio voi olla hyvin monimutkainen, sillä se voi esimerkiksi kysyä joltain järjestelmältä lisätietoa, muuttaa sanomaa sen perusteella, ja generoida uusia sanomia. Orkestraatio kirjoittaa kaikki sanomat messageboxiin. Messageboxista sanomat menevät määriteltyihin send-portteihin niiden tilausten perusteella. (Tarhonen 2014b.)

Send-porttien kautta sanomat lähtevät ulos Biztalkista. Vastaavaan tapaan kuin receive locationissa, send-porttiin liittyy pipeline, mahdolliset pipelinen lisäkomponentit, mappaukset ja adapteri. Send-portteja voidaan ryhmitellä ryhmiin (Send Port Group). Send-porttien ryhmä tilauksen avulla sama sanoma voidaan lähettää ulos monen send-portin kautta. (Tarhonen 2014b.)

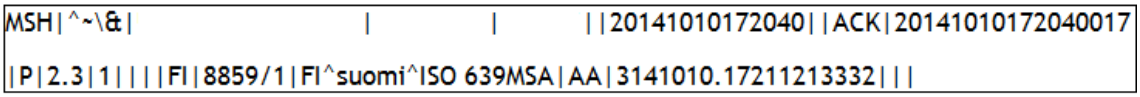
Kullekin receive locationille, orkestraatiolle ja send-portille määritellään looginen host, joka vastaa varsinaisesta suorittamisesta. Hostin alle on määritelty useita host instansseja eli Windowsin palveluja, joita ajetaan eri koneissa. Tällä menetelmällä saavutetaan suorituskykyä ja vikasietoisuutta, sillä useat koneet vastaavat tehtävän suorittamisesta. Host instansseja ajetaan sovelluspalvelimissa. Messageboxi sijaitsee omassa SQL-klusterissaan, joka koostuu useista palvelimista. Sanomakäsittelyä suorittavat sovelluspalvelimet (host instanssien ajoalustat), jotka keskustelevat keskenään messageboxin kautta. (Tarhonen 2014b.)

## 3.2 Sanoman rakenne HL7–standardin pohjalta

Sanomia voidaan tuoda Biztalkiin monilla eri standardeilla, kuten esimerkiksi yllä olevassa luvussa mainituilla HL7, EDIFACT tai XML. Tässä luvussa käsitellään sanoman rakennetta HL7 -standardin pohjalta, jota käytetään laajalti varsinkin terveydenhuollon sanomaliikenteessä. Alun perin 1987-luvulla kehitetystä HL7-standardista on syntynyt merkittävä globaali standardi terveydenhuollon yrityksille tiedon vaihtamisen välineenä. HL7 ei ole kuitenkaan sidottu vain terveydenhuoltoon, vaan teknisessä mielessä muuntamalla se toisenlaiseen formaattiin, se soveltuu myös muille toimialoille. (Henderson 2003, 3,14,24.)

HL7-sanoma jaetaan tietoryhmiin eli segmentteihin, jotka kukin sisältävät tiettyä standardin mukaista tietoa. Segmentit jaetaan vielä pienempiin kokonaisuuksiin, kuten kenttiin, kentät komponentteihin, ja komponentit osakomponenteiksi. Tämä alla oleva sanoma sisältää yhden MSH-segmentin eli otsikon, joka sisältyy jokaiseen HL7-sanomaan (Kuva 7). Muita segmenttejä voisi olla EVN (Event Information), PID (Patient Identification), PV1 (Patient Visit Information), OBX (Observation Information), AL1 (Allergy Information), jotka alkaisivat uudelta omalta riviltään. Suurin osa HL7-sanomista koostuu useista yllä mainituista segmenteistä. Segmentin tunnisteen (MSH) jälkeen määritellään erotin, jolla erotetaan segmentin sisältävät kentät toisistaan. Erottimena toimii neljännestä merkistä alkava pystyviiva eli "|". Oletusarvoiset encoding-merkit (^~\&) määritellään MSH-segmentissä kerran seuraavassa alkavassa kentässä heti pystyviivan jälkeen. Ensimmäisenä määritellään komponenttien erotin eli "^", jolla on eroteltu muun muassa komponentit "FI^suomi". Seuraavana repetition "~", jolla ilmoitetaan jonkin kentän toistuminen. Kolmantena määritellään escape "\", jota tarvitaan silloin, kun varsinainen tieto sisältää erotinmerkkejä. Viimeisenä määritellään

osakomponenttien erotin eli ”&”. Segmenttien erottimena toimii cr (Carriage Return) eli rivinvaihto. (Henderson 2003, 25-29; Tarhonen 2014a, 12.)




Kuva 7. Esimerkki HL7-sanoman rakenteesta (organisaatioiden nimet on sensuroitu pois)

Yllä olevassa sanomassa on esimerkki toisesta organisaatiosta saadusta kuittauksesta (ACK = ACKnowledgment) edelliseen lähetettyyn sanomaan (Kuva 7). Tällä tavoin sanomaliikenteessä varmistetaan, että sanoma on mennyt perille onnistuneesti. HL7-sanoman sisältämiin kenttiin on etukäteen määritelty, minkälaista tietoa niihin saa laittaa. Monia kenttiä on jäänyt tyhjäksi, jotka huomataan peräkkäisistä pystyviivoista. Jos kenttään laitettavaa tietoa ei tarvita tai käytetä, se jätetään tyhjäksi. Esimerkiksi encoding-merkkien neljään seuraavaan kenttään on määritelty lähettävä sovellus, lähettävä organisaatio, vastaanottava sovellus ja vastaanottava organisaatio, jotka on tässä tapauksessa sensuroitu pois sanomasta. Sanomasta kuitenkin nähdään sen pääpiirteet. Siitä huomataan seuraavien kenttien sisältävän ajan (20141010172040), sanomatyypin (ACK), sanoman tunnisteen (20141010172040017), käsittelyn ID (P = Production), version ID (2.3), version järjestysnumeron (1), maakoodin (FI), merkistönumeron (8859/1), ja sanoman pääkielen (FI^SUOMI^ISO 639). MSH-segmentin jälkeen sanomassa on MSA-segmentti, joka sisältää kuittauskoodin (AA) ja kuitattavan sanoman tunnistenumeron (3141010.17211213331). (Corepoint Health 2013; Tarhonen 2014a, 11.)

## 3.3 Virhetietojen muodostus tapahtumalokista

Biztalkin komponentit kirjoittavat tietoja virheellisistä sanomista Biztalkin tapahtumalokiin. Lokimerkinnän tyyppi voi olla Critical, Warning, Error, Information, Audit Success tai Audit Failure. Virheelliset sanomat (Cricital, Warning, Error) kootaan Windowsin lokijäsentimellä tapahtumalokista HTML-tiedostoon (Liite 2). Lokimerkinnän sisällöstä otetaan vain tarpeelliset tiedot mukaan, kuten dokumentin alussa määritetyt kentät (Time, Event Type, Computer, Source Name, Event Id, Message). Seuraavalla sivulla olevassa kuvassa on kuvattu kentät yksittäisestä virheellisestä sanomasta (Kuva 8), jotka on otettu Biztalkin tapahtumalokista (Liite 4).

```
Source:        BizTalk Server
Date:          10/9/2014 6:17:16 PM
Event ID:      5743
Level:         Warning
Computer:      XXXXXXXX10.XXX.FI
Description:
The adapter failed to transmit message going to send port
"TO_XXXX_OW_HTTP" with URL
"https://XX.XXXXXXXXXXXX.XX.com/XXXXXXX/XXXXXXXXXX/XXXHTTPReceive.dll?XX".
It will be retransmitted after the retry interval specified for this Send Port.
Details:"The underlying connection was closed: An unexpected error occurred on a send.".
```

Kuva 8. Esimerkki tiedoista, jotka poimitaan lokijäsentimellä Biztalkin tapahtumalokista virhelokiin (organisaatioiden nimet ja IP-osoitteet on sensuroitu pois)

Yllä oleva kuva näyttää ne kentät tapahtumalokista, joista poimitaan tiedot virhelokiin (Kuva 8). Alla olevassa kuvassa on taas samat tiedot HTML-muodossa virhelokissa määritettyjen kenttien mukaisesti (Kuva 9).

```
<TR bgCOLOR="#C0C0C0">
  <TD><tt>9.10.2014 06:17:16</tt></TD>
  <TD><tt>Warning event</tt></TD>
  <TD><tt>XXXXXXXX10.XXX.fi</tt></TD>
  <TD><tt>BizTalk Server</tt></TD>
  <TD><tt>5743</tt></TD>
  <TD><tt>The adapter failed to transmit message going to
 send port "XX_XXXXXXXXX-XXX-XXX_OW_MLLP" with URL
"XX.XXX.XXX.XXX:XXXXX". It will be retransmitted after the
 retry interval specified for this Send Port. Details:"The
 underlying connection was closed: An unexpected error
occured on a send.". </tt></TD>
</TR>
```

Kuva 9. Virhetapahtumat jäsenneltynä virhelokissa (organisaatioiden nimet ja IP-osoitteet on sensuroitu pois)

# 4 Sovelluksen esittely

Tässä luvussa käydään läpi kehitettyä sovellusta ja selostetaan sen pääpiirteet siinä järjestyksessä, jonka mukaan ne tehtiin. Sovelluksen toteutus alkoi syksyllä 2013 (Liite 1), kun sain Tietotarhan toimitusjohtajalta luettavakseni tiedoston (Liite 2), joka sisälsi tietoja sanomien virheistä. Huomasin sen sisältävän satoja hieman samantyyppisiä virheitä pääasiallisesti Warning ja Error -tapahtumia. Sain tehtäväkseni lukea tiedoston sisältämät sanomien virhetiedot Caché-tietokantaan Caché Objectscript -ohjelmointikielellä, jonka harjoittelun olin aloittanut jo hieman aiemmin kesällä. Tämän ensimmäisen varsinaisen hyödyllisen sovelluksen tarkoituksena oli myös aloittaa Cachén tutustuminen, ja kartuttaa Caché-osaamista tulevaisuuden varalle. Cachén käyttö tulee riippumaan täysin asiakkaiden

käyttämistä ohjelmistoista. Toisaalta saattaa käydä niin, ettei Cachéta valita mihinkään ja osaamiseni valuu hukkaan. Projektin aikana olen kuitenkin oppinut erittäin paljon sanomaliikenteestä ja Caché-ohjelmoinnista vuoden aikana, mitä esittelen seuraavissa kappaleissa.

## 4.1 Sovelluksen määrittely

Ennen ohjelmoinnin aloittamista sovellusta varten luotiin vaatimusmäärittely. Toimeksiannon mukaan sen pitää pystyä yhden istunnon aikana lukemaan ennalta määritetystä hakemistopolusta kaikista mahdollisista tiedostoista tietokentät Caché-tietokantaan. Sen pitää lukea vain HTML-loppuisia tiedostoja, koska kaikki virhelokitiedostot ovat HTML -muodossa. Tällä tavalla hakemistopolkuun eksyneet vääränlaiset tiedostot eivät vahingossa aukea istunnon aikana ja syö tarpeettomasti muistia. Sen pitää osata tarkistaa puutteet tai virheet jokaisen tiedoston kohdalla sen sisältävistä kentistä (Liite 2), jotka on määritelty tiedoston alussa (Time, Event Type, Computer, Source Name, Event Id, Message).

Tärkeimmän asian päätettiin olevan virheen tapahtumisaika (Time), jonka täytyy löytyä jokaisesta virheellisestä sanomasta. Jos aikaa ei löydy, sovellus lopettaa virhelokitiedoston tarkastamisen ja palauttaa virheen, kunnes se on korjattu. Tämä siksi, että jos muita virheellisiä kenttiä löytyy jostain syystä tiedostosta, pystytään ajan avulla paikantamaan tapahtumalokista kyseinen virheellinen tapahtuma helposti. Myös näiden virheellisten kenttien vuoksi, sovelluksen täytyy pystyä kirjoittamaan erilliseen ErrorLog –nimiseen tiedostoon istunnon aikana tapahtuneet virheet. Sovelluksen käyttäjän pitää pystyä löytämään ErrorLogista tarvittavat tiedot siitä, missä virhe tapahtui, jotta se voitaisiin manuaalisesti korjata. Hakemistopolku tulee todennäköisesti sisältämään vuorokausittain luotuja virhelokitiedostoja, joten ne täytyy joko siirtää pois kansiosta tai nimetä uudelleen, ettei samoja virhelokitiedostoja tallennettaisi uudelleen tietokantaan. Muut eksyneet tiedostot pitää myös siirtää pois, jotta hakemistopolku pysyy puhtaana.

Sovelluksen ajon jälkeen tietokannan tietoja pitäisi pystyä tarkastelemaan selaimessa. Tätä varten joudutaan hankkimaan tietoa Caché Zenistä, jonka avulla tietokannasta saadaan tuotua tiedot selaimessa aukeavaan taulukkoon. Taulukon pitää pystyä ensisijaisesti järjestelemään kaikkia Caché-tietokantaan sovelluksen avulla tallennettuja tietoja. Biztalkissa tapahtumaloki ei ole riittävä tällaiselle sanomien nopealle tarkastelulle.

## 4.2 Sovelluksen toteutus

Sovellusta lähdettiin kehittämään Cachén mukana tulevalla Caché-studio-sovelluskehittimellä. Ensimmäisen osan sovelluksesta, jota tullaan myös tämän opinnäytetyön jälkeen

jatkokehittämään eteenpäin, suunniteltiin toimimaan manuaalisesti Caché-terminaalissa, josta pääsääntöisesti Cachélla tehtyjä sovelluksia ajetaan.

Ensimmäiseksi käytetään Cachén mukana tulevaa luokkien tekemiseen käytettävää aputoimintoa (Class Wizard), jonka avulla sovellukselle määritellään ensimmäisellä rivillä paketin nimi, sovelluksen nimi ja luokka (Liite 1). Ensimmäinen osa nimestä ”ZenApp1” kertoo paketin (Package) nimen, jonka avulla nivotaan yhteen kaikki sovellukseen liittyvät luokat yhden kokonaisuuden alle. Tämän alle kuuluvat liitteissä kuvatut luokat, kuten Caché Objectscriptillä tehty sovellus, Caché Zenillä tehty taulukko ja Caché Zen Application –luokka (Liite 1, 3, 5). Seuraava osa nimestä ”App1” kertoo itse sovelluksen nimen ja suluissa olevat ”(%Persistent, %XML.Adaptor)” määrittelevät luokan. Tämä luokka tallentaa tiedot pysyvästi levylle sekä käyttää hyväksi XML-adapteria, jota tarvitaan tietojen saamiseksi taulukkoon.

%Persistent –luokka määritystä käytetään levylle tallennettavien olioiden (Objects) kanssa. Jokaisesta virhelokissa olevasta virheellisestä tapahtumasta tehdään olio. Tällöin ne tallentuvat Caché-tietokantaan yhtenä loogisena asiakokonaisuutena alussa määritettyjen kenttien perusteella (Time, Event Type, Computer, Source Name, Event Id, Message).

Sovelluksen pääohjelman teko aloitetaan luokkametodilla ”ClassMethod”, joka nimettiin ensimmäisellä asialla, jonka sovellus pitäisi tehdä ”lueTiedosto”. Tämän sisälle rakennetaan koko ajettava sovellus. Pääohjelman jälkeen suurin osa koodista on pilkottu aliohjelmiksi, jotka muun muassa näyttävät kaikki tiedostot ennalta määritetystä hakemistopolusta, siirtävät vääränlaiset tiedostot pois hakemistopolusta, lukevat yksitellen avatut tiedostot, ja tallentavat ne tietokantaan olioina jne. Aliohjelmat alkavat pääohjelman sisällä aina vasemmasta laidasta (Liite 1). Ensimmäinen aliohjelma alkaa sivulla kaksi nimeltä ”showFiles(dir), jonka alle on kirjoitettu toiminto, joka vastaanottaa ennalta määritetyn hakemistopolun ja palauttaa takaisin sen sisältämät HTML-tiedostot taulukkona. Seuraavissa alaluvuissa esitellään sovelluksen pääpiirteet koodinpaloilla.

### 4.2.1 Tiedostojen nimien selvittäminen hakemistopolusta

Ensimmäinen asia, joka jouduttiin selvittämään luokan määrityksen jälkeen oli, miten saadaan haltuun jostain hakemistopolussa sijaitsevien tiedostojen nimet, jotta ne saataisiin avattua yksi kerrallaan. Yhden ennalta tiedetyn tiedoston avaaminen olisi helppoa, mutta monien tuntemattomien tiedostojen haku ja avaaminen tuotti vaikeuksia jo alkumetreillä. Tässä kohtaa jouduin kysymään apua kokeneemmalta henkilöltä yrityksessämme. Pulmaan löydettiin ratkaisu seuraavalla tavalla.

```
showFiles(dir)
    set rs = ##class(%ResultSet).%New("%Library.File:FileSet")
    do rs.Execute(dir, "*", "Type")
```

Kuva 10. Ennalta määritetyn hakemistopolun avaaminen

Yllä olevassa koodissa sovellus asettaa muuttujaan ”rs” luokkakirjastosta poimitun ”%ResultSet” -luokan, joka tarjoaa tavan käyttää luokkien hakua (Class Query) sovelluksesta käsin (Kuva 10). Seuraavaksi sovellus avaa uuden ilmentymän ”.%New” metodilla luokasta ”%Library.File”, joka edustaa tiedostojen tietojärjestelmäpalveluita. Komennolla ”do”, jolla kutsutaan ohjelmia Cachéssa, ajetaan muuttujaa ”rs” käyttävä komento ”Execute”. Tälle annetaan parametrina ennalta määritetty hakemistopolku ”dir”, joka on kirjoitettu heti pääohjelman alkuun ”set directory = "C:\Users\TT\Desktop\Juho\ App1Testi"”. Tästä hakemistopolusta haetaan kaikki tiedostot Windows-kansiosta tiedostopäätteen nimen mukaan ”Type”. Tällä tavoin sovellus lukee eri tiedostopäätteet järjestyksessä, esimerkiksi .html ja .txt, hakemistopolusta. Tämän jälkeen samassa aliohjelmassa käytetään ohjausrakennetta ”While”, jonka avulla lisätään HTML-päätteisiä tiedostoja taulukkoon niin kauan, kuin niitä vain löytyy hakemistopolusta. Tästä taulukosta saadut tiedostonimet käydään yksitellen lävitse seuraavassa kappaleessa. (InterSystems 2013e; InterSystems 2013f.)

### 4.2.2 Tiedostojen avaus

Aliohjelmasta ”showFiles(dir)” palataan takaisin pääohjelmaan, jossa lähdetään avaamaan tiedostoja saamasta taulukosta seuraavalla tavalla.

```
set file = ##class(%File).%New(fileNamePaths(i))
do file.Open("R",2)
```

Kuva 11. Tiedostonimien sisältävän taulukon läpikäynti ja avaaminen

Yllä olevassa koodin muuttujassa ”file” käytetään uudestaan luokkaa %Library.File”, jonka avulla luodaan uusi ilmentymä taulukosta ”fileNamePaths(i)”, mihin on tallennettu kaikki hakemistopolun HTML-loppuiset tiedostonimet (Kuva 11). Näitä tiedostoja sovellus avaa yksitellen syntaksilla ”do file.Open(”R”,2), jossa on määritelty ”Open” -komennon sisällä annetut arvot tiedoston lukemiselle (R = Read) ja kahden sekunnin viive (TimeOut). Viive kannattaa määritellä siksi, että jos avattava tiedosto ei vastaa annetun ajan sisällä, ohjelma jättää kyseisen tiedoston väliin ja siirtyy seuraavaan. Muuten koko sovellus jää odottamaan ikuisesti tiedoston avautumista. (Kadow 2012, 180.)

### 4.2.3 Rivien luku ja muokkaus

Kun tiedosto on avautunut, se viedään toiseen aliohjelmaan nimeltä ”readFile(file)”, jossa avattua tiedostoa lähdetään käymään lävitse rivi riviltä seuraavalla tavalla.

```
do readFile(file)
set line = file.ReadLine()
if (line?.E1"<TD><tt>".E1"</tt></TD>")
set line = $$editLine(line)
```

Kuva 12. Avatun tiedoston määriteltyjen rivien läpikäyminen

Yllä olevassa koodissa muuttujaan ”line” luetaan tiedoston alusta alkaen rivi kerrallaan tiedoston loppuun saakka (Kuva 12). Tiedostosta halutaan lukea tietokantaan vain ne rivit, joissa on tietoa virheellisistä sanoman tapahtumista. Tämän kohdalla käytetään hyväksi ns. merkkijonon tunnistusta (Pattern Matching), jonka avulla seuraavaan aliohjelmaan viedään editoitavaksi vain tietyt määritetyt rivit. Merkkijonon tunnistus aloitetaan kysymysmerkillä (?), jonka jälkeen merkkijono saa sisältää mitä tahansa merkkejä ”.E”, yhden ”<TD><tt>” -merkkijonon, uudestaan mitä tahansa merkkejä ”.E” ja yhden ”</tt></TD>” -merkkijonon. Näin saadaan kaikki ” <TD><tt></tt></TD>” -rivit virhelokitiedostosta, jonka alussa saa olla tiedoston asetuksen mukainen tyhjä rivi, ja itse tietoa HTML-tagien välissä. Merkkijonon tunnistuksen jälkeen rivit viedään yksitellen seuraavaan aliohjelmaan muokattavaksi syntaksilla ”set line = $$editLine(line)”. Tätä syntaksia ”$$” käytetään käyttäjän omissa määritellyissä toiminnoissa (Function), kun johonkin muuttujaan halutaan palauttaa kyseisestä aliohjelmasta tietoa. (InterSystems 2013h.)

```
editLine(line)
    set line = $piece($piece(line,">",3),"<",1)
    if (line?1.2N1"."1.2N1"."2.4N1" "1.2N1":"1.2N1":"1.2N){
        set line = $replace(line, ".", "/")
        set line = $ZDATETIME($ZDATETIMEH(line, 4), 3)
```

Kuva 13. Määritettyjen rivien muokkaus

Yllä esitettyyn aliohjelmaan ”editLine(line)” viedyistä rivistä muokataan pois kaikki muut, paitsi HTML-tagien välissä oleva tieto (Kuva 13). Tässä kohdassa sovellus käyttää jo aiemmin käsiteltyä $piece -toimintoa, jonka avulla se leikkaa esimerkiksi rivistä ”<TD><tt>Warning event</tt></TD>” alimerkkijonon ”Warning event”. Kun tähän aliohjelmaan tuodaan rivi, jossa käsitellään aikaa ”<TD><tt>7.8.2013 07:31:04</tt></TD>”, se muunnetaan sovelluksen alussa määritettyyn ”%TimeStamp” -formaattiin, joka on muotoa ” YYYY-MM-DD HH:MM:SS.nnnnnnnnn”. (InterSystems 2013g.)

### 4.2.4 Olioiden luonti ja tallentaminen tietokantaan

Tässä luvussa käsitellään sovelluksen tärkeintä ominaisuutta, eli edellisessä luvussa muokattujen rivien tallentamista olioina Caché-tietokantaan. Muokatut rivit viedään seuraavaan aliohjelmaan nimeltä “createObject(line)” (Kuva 14). Pääohjelmassa alustettuun muuttujaan ”objectLine” lisätään plus yksi $Increment –toiminnolla joka kerta, kun uusi rivi tuodaan alustettavaksi tähän aliohjelmaan. ”objectLine” –muuttujassa pidetään laskuria rivien lukumäärästä. Kuuden rivin jälkeen (DateAndTime, Event Type, Computer, Source Name, Event Id, Message) olio tallennetaan tietokantaan syntaksilla “set Save=App1Oref.%Save()”. Muuttujaan ”App1Oref” on kirjoitettu uusi tallennettava globaali nimeltään ”ZenApp1.App1”, johon oliona lisätään kuusi riviä tietoa, jonka jälkeen se tallennetaan tietokantaan. Virhelokitiedostossa olevat virheelliset tapahtumat sisältävät aina vain kuusi riviä tietoa. Tällä tavalla ohjelma luottaa hyvin paljon virhelokitiedoston oikeellisuuteen, koska jos yksikin rivi puuttuu ohjelma lähettää virheen ja keskeytyy ajan kohdalla, joka on määritelty pakolliseksi sovelluksen alussa syntaksilla “Property DateAndTime As %TimeStamp [ Required ];”.

```
set objectLine = $increment(objectLine)
if (objectLine = 1){set App1Oref=##class(ZenApp1.App1).%New()
                    set App1Oref.DateAndTime=line}
if (objectLine = 2){set App1Oref.EventType=line}
if (objectLine = 3){set App1Oref.Computer=line}
if (objectLine = 4){set App1Oref.SourceName=line}
if (objectLine = 5){set App1Oref.EventId=line}
if (objectLine = 6){set App1Oref.Message=line
                    set Save=App1Oref.%Save()
```

Kuva 14. Olioiden luonti muokatuista riveistä ja niiden tallennus Caché-tietokantaan

### 4.2.5 Tiedostojen siirto

Edellisessä luvussa muokatut rivit tallennettiin Caché-tietokantaan. Tässä luvussa käydään lävitse miten tiedostot siirretään pois hakemistopolusta, jotta niitä ei luettaisi uudelleen tietokantaan seuraavalla kerralla, kun sovellusta ajetaan.

Seuraavalla sivulla olevassa koodissa sovellus vie käytetyt tiedostonimet uudelleen nimettäväksi aliohjelmaan ”renameFile” (Kuva 15). ”%Library.File” -luokkaan sisältyvällä ”Rename” luokkametodilla alkuperäinen tiedosto ”oFileName” nimetään halutunlaiseksi ”newFileName”. Tässä tapauksessa HTML-tiedostoja ei siirretä pois määritetystä hakemistopolusta, vaan niiden perään laitetaan ainoastaan sana ”(OK)”, jolloin ne jätetään pois sovelluksen alussa laaditusta taulukosta.

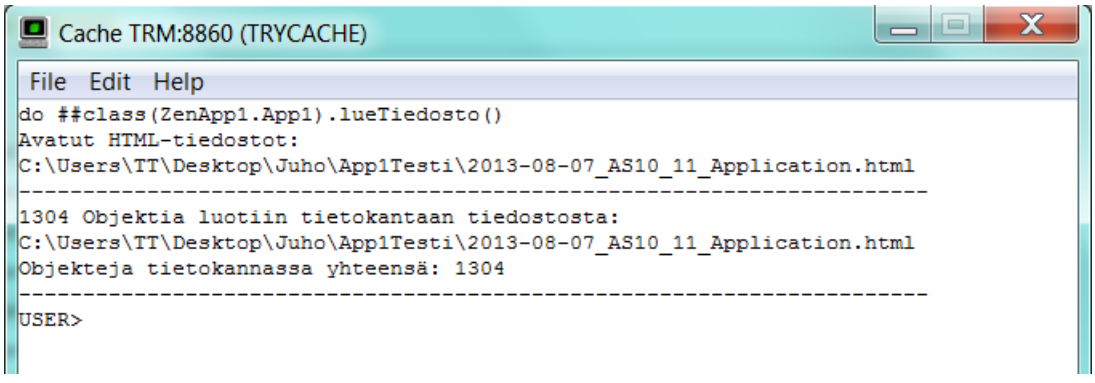
```
renameFile(oFileName, newFileName)
    set renamedFile = ##class(%File).Rename(oFileName, newFileName)
```

Kuva 15. Luettujen tiedostojen uudelleen nimeäminen

Aiemmissa luvuissa käytiin lävitse sovelluksen pääpiirteet eli tiedostojen nimien paikannus hakemistopolusta ja tallentaminen taulukkoon, tiedostojen avaaminen yksitellen taulukosta, tiedostojen rivien luku ja muokkaaminen, tietoa sisältävien rivien tallentaminen olioina tietokantaan, ja tiedostojen uudelleen nimeäminen. Sovellus katsoo monilla määritellyillä laskureilla virheitä, joita voi esiintyä virhelokissa. Se käyttää myös Caché-terminaalissa ohjelman ajon aikana käyttäjän apuna muutamia asioita, kuten esimerkiksi aliohjelmia ”viiva” ja ”pressToContinue”, joiden tarkoituksena on auttaa havainnollistamisessa. Havainnollistamisen tarkoituksena on antaa käyttäjälle mahdollisuus pysäyttää ohjelma ajon aikana Caché-terminaalissa ja erottaa näytölle tulleet rivit toisistaan. Myös harvinaisten sovelluksen virhetilanteiden tarkastelulle on luotu erillinen aliohjelma ”createErrorLog”, jota ei ole vielä jouduttu käyttämään oikeassa tilanteessa kertaakaan. Kaikkea sovelluksesta ei käyty lävitse, mutta aiemmissa luvuissa kerrottiin sen pääasiat liitteestä 1. Seuraavassa kappaleessa testataan sovellusta ja näytetään, miten se suoritetaan Caché-terminaalissa.

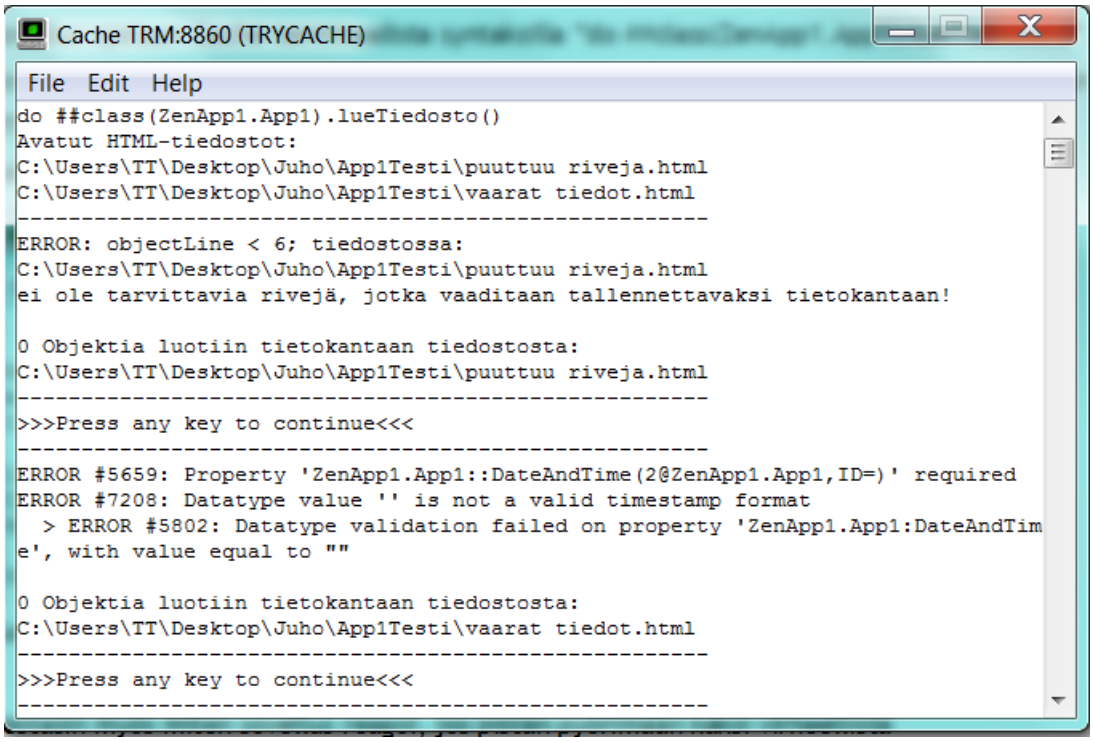
## 4.3 Sovelluksen testaus

Sovellus ajetaan Caché-terminaalista syntaksilla ”do ##class(ZenApp1.App1).lueTiedosto()”. Sovellus käynnistyy ja se toimii vaadittavalla tavalla. Alla olevassa kuvassa on virheettömän HTML-tiedoston ajo, josta huomataan että tiedosto sisältää 1304 virhetilanteisiin liittyvää lokitapahtumaa (Kuva 16).




Kuva 16. Sovelluksen ajo Caché-terminaalissa

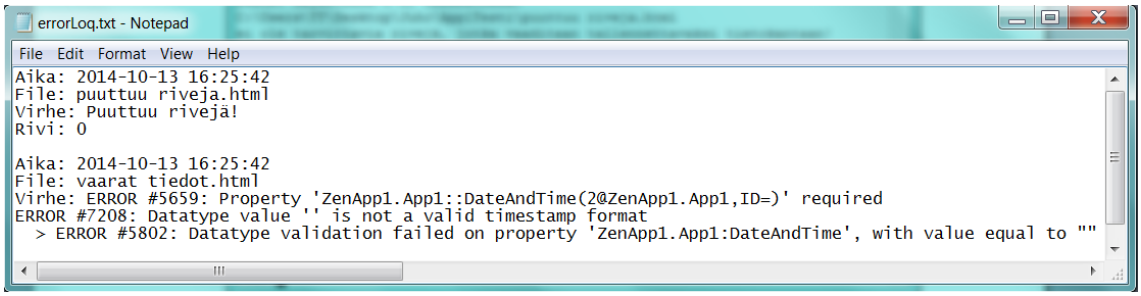
Sovellusta täytyy myös testata virheellisten tiedostojen avulla. Seuraavalla sivulla olevassa kuvassa testataan, miten sovellus reagoi, jos pistetään pyörimään kaksi virheellistä virhelokitiedostoa, joiden alusta puuttuu rivejä ja ajan kohdalla tietokenttä (Kuva 17).




Kuva 17. Virheellisten virhelokien ajo Caché-terminaalissa

Kuten yllä olevasta kuvasta huomaa sovellus palauttaa virheen tiedostosta, josta puuttuu rivejä muodossa ”ERROR: objectLine <6; tiedostossa:” (Kuva 17). Jos tietokentät sisältävät väärää tietoa, sovellus palauttaa virheen muodossa ”ERROR #7208: Datatype value " " is not a valid timestamp format”, joka vaatii oikentyyppistä tietokenttää ajan kohdalla.

Kyseiset virheet tallentuvat ErrorLogiin onnistuneesti alla olevassa muodossa (Kuva 18).




Kuva 18. ErrorLogiin tallentuneet virheet

Sovellus ottaa talteen ajan milloin virhe tapahtuu, tiedoston nimen, itse virheen ja rivin. Tässä tapauksessa mitään ei tallentunut tietokantaan, koska virhe tapahtui heti tiedoston alussa, joten rivin arvoksi muodostui 0.

## 4.4 Selaimessa toimivan taulukon luonti Caché Zenillä

Caché Zenillä saadaan luotua sovellukselle taulukko käyttäen hyväksi metodigeneraattoria (Liite 3). Tämä pelkästään taulukon sisältävä luokka nimetään ”ZenApp1.HomePage”. Ohjelman alusta voidaan huomata, että se käyttää hyväkseen ”%ZEN.Component.page” -luokkaa, jonka avulla määritellään sivun käyttämät komponentit. Taulukkosivun lisäksi joudutaan myös tekemään tyhjä luokka (Liite 5), jossa määritellään sovelluksen nimi ja sovellusta käyttävä kotisivu seuraavilla parametreilla ”Parameter APPLIOCATIONNAME = ”ZenApp1”;” ja ”Parameter HOMEPAGE = ”ZenApp1.HomePage.cls”;” (Kuva 19).

```
/// ZenApp1.Application
Class ZenApp1.Application Extends %ZEN.application
{

/// This is the name of this application.
Parameter APPLICATIONNAME = "ZenApp1";

/// This is the URL of the main starting page of this application.
Parameter HOMEPAGE = "ZenApp1.HomePage.cls";

/// This Style block contains application-wide CSS style definitions.
XData Style
 {
 <style type="text/css">
 </style>
 }

}
```

Kuva 19. Taulukkoluokan vaatima sovellusluokka

Itse taulukkoluokkaan lisätään taulukko komponentilla ”<tablePane tableName="ZenApp1.App1">”, ja taulukko yhdistetään sovellukseen ”tableName” -attribuutilla (Liite 3). Sovelluksen ensimmäisillä riveillä on ns. propertyina määritelty tietokantaan tallennettavien rivien tietotyypit (Liite 1). Ajan kohdalla se käsitellään ”%TimeStamp” -formaatissa ja muiden kohdalla ne laitetaan ”%String” -formaattiin. Ainut poikkeus oli ”EventId”, joka näkyy jokaisessa tietokentässä kokonaislukuna (Liite 2). Tämän vuoksi ajan kanssa se oli ainoa kenttä, jonka tiedetään sisältävän aina samantyyppistä tietoa. Muiden kohdalla sovellus ei voi tietää minkälaista tietoa ne voivat sisältää, joten ne tallennetaan tietokantaan merkkijonoina. ”Message” kohdan sovellus muuttaa ”%Text” -formaattiin, jotta taulukko pystyy filtteröimään tekstin seasta useita sanoja kerralla. Taulukkoon lisätään muun muassa sivun koko, rivien enimmäismäärä, taulukon leveys yms. Tärkein asia on lisätä alla olevat sarakkeet, joilla taulukko ottaa tietokannasta tiedot (Kuva 20).

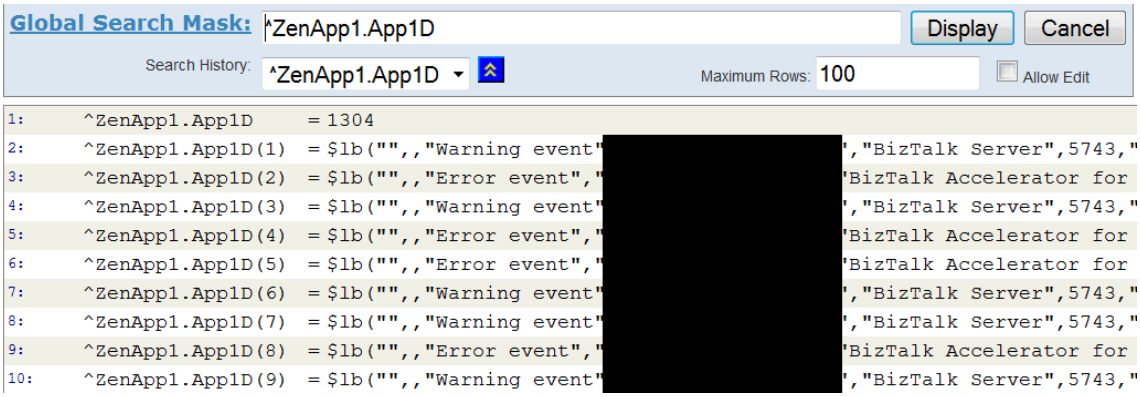
```
<column colName="ID" hidden="true"/>
<column header="Date And Time" width="" colName="DateAndTime" filterType="datetime"
filterOp="BETWEEN"/>
<column header="Event Type" width="" colName="EventType" filterType="text"
filterOp="UP["/>
<column header="Computer" width="" colName="Computer" filterType="text"
filterOp="UP["/>
<column header="Source Name" width="" colName="SourceName" filterType="text"
filterOp="UP["/>
<column header="Event Id" width="" colName="EventId" filterType="text"
filterOp="UP["/>
<column header="Message" width="" colName="Message" filterType="text"
filterOp="%CONTAINS"/>
</tablePane>
```

Kuva 20. Sarakkeet joiden avulla tietokannasta otetaan tiedot taulukkoon

Hyvin pitkälti metodigeneraattori määrittelee käyttäjälle tyylitiedot ja muut tarvittavat asiat XML-kehyksen sisälle. Nyt taulukko saa otettua tiedot tietokannasta, mutta se ei vielä osaa järjestellä niitä. Tätä varten joudutaan määrittelemään yllä olevan kuvan mukaan sarakkeiden attribuutit ”filterType” ja ”filterOp” (Kuva 20). Itse komponentti, jolla liikutaan eri sivuilla taulukkokentässä on muotoa ”<tableNavigatorBar id="App1Nav" tablePaneId="App1Table"/>”, joka sidotaan taulukkoon attribuutilla ”tablepaneId”.

## 4.5 Caché-tietokannan tietojen avautuminen selaimeen

Caché-terminaalista ajettu sovellus toimii testauksen jälkeen halutulla tavalla. Suurin osa vioista, joita käyttäjä tai virhelokin puutteelliset kentät voivat aiheuttaa, merkitään ErrorLogiin. Caché-tietokanta tallentaa tapahtumalokitiedot niin kuin sen pitäisi. Alla olevasta kuvasta nähdään, että globaali nimeltään ”^ZenApp1.App1D” sisältää 1304 tapahtumatietoa yhdestä ajetusta tiedostosta (Kuva 21).




Kuva 21. Caché-tietokantaan tallennetut tapahtumatiedot (organisaatioiden nimet on sensuroitu pois)

Tiedot aukeavat alla olevaan taulukkoon seuraavanlaisesti.

Results: 1304 Page: 1 2 3 4 5 of 14

**ZenApp1.TablePane**

| # | Date And Time | Event Type | Computer | Source Name | Event Id | Message |
|---|---|---|---|---|---|---|
| 1 | 2013-08-07 07:31:04 | Warning event | [redacted] | BizTalk Server | 5743 | The adapter failed to transmit |
| 2 | 2013-08-07 07:31:04 | Error event | | BizTalk Accelerator for HL7 | 8451 | Unable to receive ACK from n |
| 3 | 2013-08-07 07:30:09 | Warning event | | BizTalk Server | 5743 | The adapter failed to transmit |
| 4 | 2013-08-07 07:30:09 | Error event | | BizTalk Accelerator for HL7 | 8451 | Unable to receive ACK from n |
| 5 | 2013-08-07 07:21:04 | Error event | | BizTalk Accelerator for HL7 | 8450 | Unable to write message to n |
| 6 | 2013-08-07 07:21:04 | Warning event | | BizTalk Server | 5743 | The adapter failed to transmit |
| 7 | 2013-08-07 07:18:09 | Warning event | | BizTalk Server | 5743 | The adapter failed to transmit |
| 8 | 2013-08-07 07:18:09 | Error event | | BizTalk Accelerator for HL7 | 8451 | Unable to receive ACK from n |
| 9 | 2013-08-07 07:12:07 | Warning event | | BizTalk Server | 5743 | The adapter failed to transmit |
| 10 | 2013-08-07 07:12:07 | Error event | | BizTalk Accelerator for HL7 | 8450 | Unable to write message to n |

Kuva 22. Kuva selaimeen avautuvasta taulukosta (organisaatioiden nimet on sensuroitu pois)

Taulukko näyttää sata sanomaa yhdellä sivulla, ja sen sisältämiä sanomia pystytään järjestelemään kenttien (Date And Time, Event Type, Computer, Source Name, Event Id, Message) alapuolella olevista filttereistä (Kuva 22). Lisäksi vasemmassa ylälaidassa olevan navigaatiopalkin avulla pystytään liikkumaan toisille taulukon sivuille. Sivuja näyttää olevan 14 kappaletta, kukin sisältää enintään sata sanomaa.

## 4.6 Sovelluksen ensimmäisen vaiheen lopputulos

Sovellus toimii tällä hetkellä niin kuin sen pitääkin. Virheitä koodista ei ole löytynyt ja se vastaa sille määritetyt tavoitteet. Toimeksiantaja on tyytyväinen sen toimivuuteen ja käyttöliittymään. Seuraavalla sivulla olevassa taulukossa on yhden vuorokauden aikana (24 h) muodostuneet tapahtumalokitiedot virheellisistä tapahtumista (Taulukko 2). Taulukon perusteella pystytään filtteröimään 1304 eri lokitapahtumasta muun muassa Error- ja Warning -tilanteiden lukumäärät. Error-tilanteita on 706 kappaletta ja Warning-tilanteita 598 kappaletta. Tästä voidaan päätellä, että virhelokitiedostot sisältävät enimmäkseen vain kyseisiä virheellisiä tilanteita, jotka on otettu lokijäsentimellä Biztalkin tapahtumalokista. Yhtäkään Critical-tilannetta ei löytynyt virhelokista. Alla olevaan taulukkoon poimitaan ”Event Id” -kentän perusteella kaikkien virheiden lukumäärä.

| Event Id: | Lukumäärä: | Virheen syy: |
|---|---|---|
| 1001 | 1kpl | Adapterin yhteysvirhe. |
| 1309 | 9kpl | Tapahtuman pyyntö keskeytettiin. |
| 4103 | 181 kpl | Sanomalla on viallinen MSH eli otsikkokenttä. |
| 5719 | 25 kpl | Receive pipelinen suorituksessa tapahtui virhe. |
| 5740 | 152 kpl | FTP-adapterissa tapahtui virhe. |
| 5743 | 437 kpl | Adapteri epäonnistui sanoman lähetyksessä vastaanottajalle. |
| 5749 | 38 kpl | Vastaanottaja ei vastannut pyyntöön(TimeOut). |
| 5754 | 38 kpl | Vastaanottaja ei tarjonnut riittävää vastausta. |
| 8450 | 52 kpl | Host-palvelimen yhteys lopetettiin. |
| 8451 | 284 kpl | Vastaanottaja ei lähettänyt kuittausta |
| 8453 | 49 kpl | Yhteys vastaanottavaan palvelimeen hylättiin. |
| 10034 | 38 kpl | XLANG-moottorin tapahtumaloki: ”Uncaught exception” |

Taulukko 2. Taulukko kaikista virhelokin tapahtumien lukumääristä

”Event Id” (5743) perusteella huomaa virhelokin sisältävän suurimmaksi osaksi adapterin lähetysvirheitä 437 kappaletta. Virhe (8451) ”vastaanottaja ei lähettänyt kuittausta” esiintyy yllättävän monta kertaa, joita on virhelokissa 284 kappaletta. Näitä tilanteita tapahtuu silloin, kun sanoman vastaanottaja ei vastaa tietyn ajan sisällä lähetettyyn sanomaan. Sanoman vastaanottaja lähettää lähetettyyn sanomaan takaisin kuittauksen, jota tässä tapauksessa ei ole tapahtunut määritellyn ajan sisällä. Lisäksi taulukosta huomaa sen sisältävän paljon virheellisiä HL7-sanomia (4103), joiden MSH eli otsikkokentässä on virheellisesti määriteltyjä kenttiä. Näitä virheitä on 181 kappaletta.

## 4.7 Sovelluksen jatkokehitys

Sovelluksen jatkokehitys on jo päässyt alkamaan, kun tätä opinnäytetyötä kirjoitetaan loppuun. Sovellus tullaan automatisoimaan lähes kokonaan. Tällä hetkellä sovellus toimii vain manuaalisesti, kun käyttäjä ajaa sen Caché-terminaalista. Sovelluksen pitäisi pystyä ajamaan automaattisesti jonkin tietyn ajan välein, jotta käyttäjän tarvitsee vain laittaa virhelokitiedostot ennalta määriteltyyn hakemistopolkuun. Sen pitäisi pystyä poistamaan vanhoja virheellisiä sanomatietoja pois tietokannasta käyttäjän antaman päiväyksen perusteella. Esimerkiksi kolme kuukautta vanhat sanomat poistuisivat kerran päivässä tietokannasta ja tiedostot hakemistopolusta. Sovellukselle pitää kehittää erillinen kotisivu ja taulukkosivu pitää nimetä uudelleen. Kotisivulta pitäisi pystyä ajamaan ohjelma manuaalisesti napin painalluksella ja katsomaan sovelluksen ajo ilman, että Caché-terminaalia pitäisi käynnistää. Lisäksi linkit taulukkosivulle ja sanomien poistamiselle tarkoitettu nappula.

Tämän projektin ensimmäinen osa päättyy tähän opinnäytetyöhön, jolloin sovellus toimii sellaisenaan virheettömästi. Projektin ensimmäisen vaiheen jälkeen alkaa seuraava vaihe. Seuraava vaihe jatkuu, kunnes osaan Cachéta tarpeeksi, ja sovellus on hyväksyttävässä kunnossa, jotta sen käyttäminen olisi mahdollisimman helppoa Tietotarhan henkilökunnalle.

# 5 Johtopäätökset

Tutkimuksessa saatiin vastaukset sille asetettuihin tutkimuskysymyksiin. Projektin aikana saatiin aikaiseksi toimiva tilastointisovellus, jonka ensimmäiseen vaiheeseen sisältyy Caché -terminaalista manuaalisesti ajettava ohjelma. Sovellus lukee Caché-tietokantaan onnistuneesti Biztalk-palvelimen tuottamia sanomaliikenteen virhelokitiedostoja. Virhelokien sisältämät virheelliset tapahtumat saadaan Caché-tietokannasta onnistuneesti selaimeen tarkasteltavaksi. Selaimessa olevista tiedoista voidaan katsoa tapahtuma ID:n (Event ID) perusteella virhetilanteiden lukumääriä. Yleisimmät virhetilanteet, jotka löytyvät ajetusta virhelokista, ovat muun muassa adapterin yhteysvirheitä, vastaanottajan kuittaus virheitä ja viallisia MSH-kenttiä.

Virheelliset sanomatiedot saadaan avattua HTML-taulukkoon, kun niitä varten on ensin luotu erillinen taulukkoluokka Caché Zeniä hyväksikäyttäen. Tiedot saadaan HTML-taulukkoon käyttämällä Caché Zenin mukana tuotua komponenttiluokkaa, jonka avulla se voi noutaa tietoja Caché-tietokannasta. Taulukkoluokkaan saatiin myös liitettyä ominaisuutena navigaatiopalkki ja hakukentät, joiden avulla voidaan selata ja järjestellä virheellisiä tapahtumatietoja selaimessa.

Työssä kartoitetaan Cachéta ja sen mukana tuotua Caché Objectscriptin -ohjelmointikieltä. Sen vahvuuksiin kuuluu esimerkiksi kyky käsitellä merkkijonoja sen mukana tuoduilla toiminnoilla ja tietokantana se pystyy ajamaan tuhansia prosesseja samanaikaisesti. Cachéta kritisoidaan sen nopeasta Kill-komennosta, jonka avulla voidaan hetkessä tuhota tietokannan sisältämät tiedot. Lisäksi työssä selvitetään sanoman rakenne HL7-standardin pohjalta.

Caché Objectscriptillä ohjelmointi on haastavaa aloittelevalle ohjelmoijalle, jos lähteenä käytetään pääsääntöisesti InterSystemsin dokumentaatiota. Nollapohjalta aloitettu kehittämishanke saatiin käyntiin kunnolla heti sen jälkeen, kun sain käsiini Caché ObjectScript and MUMPS –kirjan, jossa kuvataan esimerkkien avulla COS-ohjelmointikieltä. Caché-tietokanta on itsessään erittäin laaja kokonaisuus, ja sen täysimääräiseen ymmärtämiseen saattaa kulua paljon aikaa. Tässä tapauksessa sovelluksen ensimmäinen ohjelmointi vaihe kesti yli puoli vuotta aloituksesta.

Sovelluksen ensimmäisen vaiheen kehityksessä onnistuttiin kohtalaisen hyvin. Projektin aikana sovellusta oltaisiin voitu kehittää nopeammalla aikataululla, joka tässä tapauksessa venyi liikaa. Aloittelevana ohjelmoijana tämä projekti antoi hyvän kuvan siitä, mitä itsenäinen ohjelmointi vaatii käyttäjältä. Sanomien virheellisten tapahtumien osalta, sovellus olisi voitu myös säätää erikseen ottamaan vastaan myös muita, kuin virheellisiä tapahtumia.

Ennen kuin kehitetty sovellus on täysin valmis, sen täytyy toimia automaattisesti. Jatkokehitys on lähtenyt käyntiin keväällä 2014, jonka aikana sovellukselle on tehty lisää aliohjelmia, joiden avulla voidaan esimerkiksi poistaa tietoja tietokannasta. Tässä toisessa kehitysvaiheessa, joka kestää niin kauan kunnes sovellus on valmis, pyritään saamaan sovellus kokonaan vaadittavaan kuntoon, ja käyttöliittymä hiottua käyttäjäystävällisemmäksi. Toinen kehitysvaihe pitäisi pysyä ajallisesti mieluiten samoissa mitoissa, kuin ensimmäisen vaiheen, eli noin puoli vuotta.

Tähän opinnäytetyöhön saatiin koottua hyvin tietoa Caché-tietokannasta ja Caché Objectscript –ohjelmointikielestä. Työhön olisi voitu sisällyttää enemmän tietoa Caché Zenistä ja MUMPSista. Caché Objectscript –ohjelmointikielen vahvuuksia ei pystytty vertaamaan muihin ohjelmointikieliin, koska aloittelevana ohjelmoijana minulla ei ole tarpeeksi tietotaitoa niistä. Tietotarhan toimitusjohtajan haastattelulla saatiin selkeä kuva sanomaliikenteestä, sanoman eri vaiheista ja rakenteesta Biztalk-palvelimen sisällä.

Lähteet

Kirjat

Henderson, M. 2003. HL7 Messaging. United States of America: OTech.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2010. Tutki ja kirjoita. 15.-16. painos. Hämeenlinna: Kariston kirjapaino.

Kadow, P. 2012. Caché ObjectScript and MUMPS. CreateSpace.

Mohr, S. & Woodgate, S. 2000. Professional BizTalk. Canada: Wrox Press.

Sähköiset lähteet

Anttila, P. 2008. Onko opinnäytetyöstä kehittämistyöksi? Mistä löytyvät opinnäytetyön metodologiset ratkaisut?. Viitattu 17.10.2014.
https://www.chydenius.fi/pdf/anttilan-kalvot

Corepoint Health. 2013. HL7 MSH – Message Header. Viitattu 21.9.2014.
http://www.corepointhealth.com/resource-center/hl7-resources/hl7-msh-message-header

Internet FAQ Archives. 1999a. M Technology and MUMPS Language FAQ, Part 1/2. Viitattu 17.9.2014.
http://www.faqs.org/faqs/m-technology-faq/part1/

Internet FAQ Archives. 1999b. M Technology and MUMPS Language FAQ, Part 2/2. Viitattu 17.9.2014.
http://www.faqs.org/faqs/m-technology-faq/part2/

InterSystems. 2000. Database Concepts and Database Files. Viitattu 17.9.2014.
http://docs.intersystems.com/cache41/prg/prgdbconceptsfiles.html

InterSystems. 2013a. Global Structure. Viitattu 19.9.2014.
http://docs.intersystems.com/ens20102/csp/docbook/DocBook.UI.Page.cls?KEY=GGBL_structure

InterSystems 2013b. Introducing Zen. Viitattu 21.9.2014.
http://docs.intersystems.com/cache20122/csp/docbook/DocBook.UI.Page.cls?KEY=GZEN_intro

InterSystems. 2013c. Introduction to Caché Programming. Viitattu 21.9.2014.
http://docs.intersystems.com/cache20122/csp/docbook/DocBook.UI.Page.cls?KEY=GORIENT_ch_intro

InterSystems. 2013d. $LENGTH. Viitattu 17.9.2014.
http://docs.intersystems.com/cache20122/csp/docbook/DocBook.UI.Page.cls?KEY=RCOS_flength

InterSystems. 2013e. %Library.File. Viitattu 21.9.2014.
http://docs.intersystems.com/cache20122/csp/documatic/%25CSP.Documatic.cls?PAGE=CLASS&LIBRARY=%25SYS&CLASSNAME=%25Library.File

InterSystems. 2013f. %Library.ResultSet. Viitattu 21.9.2014.
http://docs.intersystems.com/cache20122/csp/documatic/%25CSP.Documatic.cls?APP=1&LIBRARY=%25SYS&CLASSNAME=%25Library.ResultSet

InterSystems. 2013g. %Library.TimeStamp. Viitattu 21.9.2014. http://docs.intersystems.com/cache20122/csp/documatic/%25CSP.Documatic.cls?APP=1&LIBRARY=%25SYS&CLASSNAME=%25Library.TimeStamp

InterSystems. 2013h. Operators And Expressions. Viitattu 21.9.2014. http://docs.intersystems.com/cache20122/csp/docbook/DocBook.UI.Page.cls?KEY=GCOS_operators#GCOS_operators_pattern

InterSystems. 2013i. $PIECE. Viitattu 17.9.2014. http://docs.intersystems.com/cache20122/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_d_piece

InterSystems. 2013j. The Caché DataBase Engine. Viitattu 17.9.2014. http://docs.intersystems.com/cache20122/csp/docbook/DocBook.UI.Page.cls?KEY=GIC_dbengine

InterSystems. 2013k. The Caché DataBase Engine. Viitattu 17.9.2014. http://docs.intersystems.com/ens20102/csp/docbook/DocBook.UI.Page.cls?KEY=GIC_DBEngine

InterSystems. 2013l. What Is Caché?. Viitattu 17.9.2014. http://docs.intersystems.com/cache20122/csp/docbook/DocBook.UI.Page.cls?KEY=GIC_intro

Virtuaaliammattikorkeakoulu. 2007a. Tutkimuksen reliabiliteetti. Viitattu 26.10.2014. http://www2.amk.fi/digma.fi/www.amk.fi/opintojaksot/0709019/1193463890749/1193464185783/1194413792643/1194415307356.html

Virtuaaliammattikorkeakoulu. 2007b. Tutkimuksen validiteetti. Viitattu 26.10.2014. http://www2.amk.fi/digma.fi/www.amk.fi/opintojaksot/0709019/1193463890749/1193464185783/1194413809750/1194415367669.html

Julkaisemattomat lähteet

Tarhonen, T. 2014a. Biztalk. Powerpoint-esitys. Viitattu 7.10.2014.

Tarhonen, T. 2014b. Ylläpitopalaveri & toimitusjohtajan haastattelu 3.10.2014. Tietotarha. Helsinki.

Kuvat

Kuva 1. Toiminnon $Length -kuvaus ..... 12
Kuva 2. Toiminnon $Piece -kuvaus ..... 12
Kuva 3. Esimerkki taulukoidusta globaalista (organisaatioiden nimet on sensuroitu pois) . 14
Kuva 4. Esimerkki Kill-komennosta ..... 16
Kuva 5. Esimerkki paljaasta viittauksesta globaaliin ..... 16
Kuva 6. Biztalk-palvelimen arkkitehtuuri (Tarhonen 2014a, 42.) ..... 18
Kuva 7. Esimerkki HL7-sanoman rakenteesta (organisaatioiden nimet on sensuroitu pois) 20
Kuva 8. Esimerkki tiedoista, jotka poimitaan lokijäsentimellä Biztalkin tapahtumalokista virhelokiin (organisaatioiden nimet ja IP-osoitteet on sensuroitu pois) ..... 21
Kuva 9. Virhetapahtumat jäsenneltynä virhelokissa (organisaatioiden nimet ja IP-osoitteet on sensuroitu pois) ..... 21
Kuva 10. Ennalta määritetyn hakemistopolun avaaminen ..... 24
Kuva 11. Tiedostonimien sisältävän taulukon läpikäynti ja avaaminen ..... 24
Kuva 12. Avatun tiedoston määriteltyjen rivien läpikäyminen ..... 25
Kuva 13. Määritettyjen rivien muokkaus ..... 25
Kuva 14. Olioiden luonti muokatuista riveistä ja niiden tallennus Caché-tietokantaan ..... 26
Kuva 15. Luettujen tiedostojen uudelleen nimeäminen ..... 27
Kuva 16. Sovelluksen ajo Caché-terminaalissa ..... 27
Kuva 17. Virheellisten virhelokien ajo Caché-terminaalissa ..... 28
Kuva 18. ErrorLogiin tallentuneet virheet ..... 28
Kuva 19. Taulukkoluokan vaatima sovellusluokka ..... 29
Kuva 20. Sarakkeet joiden avulla tietokannasta otetaan tiedot taulukkoon ..... 30
Kuva 21. Caché-tietokantaan tallennetut tapahtumatiedot (organisaatioiden nimet on sensuroitu pois) ..... 30
Kuva 22. Kuva selaimeen avautuvasta taulukosta (organisaatioiden nimet on sensuroitu pois) ..... 31

Kuviot

Kuvio 1. Projektin eri vaiheet .....8

Kaaviot

Kaavio 1. Cachén elementtien kokoamisvaiheet (InterSystems 2013c.) ..... 15

Taulukot

Taulukko 1. Esimerkki yksittäisarvona ja taulukkona esitetystä paikallisesta muuttujasta ja globaalista (InterSystems 2000.) ..... 13
Taulukko 2. Taulukko kaikista virhelokin tapahtumien lukumääristä ..... 32

Liitteet

Liite 1. Tietotarhalle Caché Objectscript –ohjelmointikielellä kehitetty tilastointisovellus 42
Liite 2. Ensimmäinen sivu virhelokiraportista ..... 47
Liite 3. Caché Zen taulukko -luokka ..... 48
Liite 4. Virhetilanteen tapahtumalokitiedot Biztalkissa ..... 50
Liite 5. Caché Zen Application -luokka ..... 51

Liite 1. Tietotarhalle Caché Objectscript -ohjelmointikielellä kehitetty tilastointisovellus

```
Class ZenApp1.App1 Extends (%Persistent, %XML.Adaptor) [ Inheritance =
left ]
{

Property DateAndTime As %TimeStamp [ Required ];

Property EventType As %String;

Property Computer As %String;

Property SourceName As %String;

Property EventId As %Integer;

Property Message As %String(MAXLEN = "");

/*Tämä ohjelma avaa (kaikki) HTML-tiedostot, lukee jokaisesta 'New
W3SVC Messages in System Event Log' -tiedostosta rivit, muokkaa tietyt
lauseet ja tallentaa ne tietokantaan objekteina. Lisäksi nimeää luetut
HTML-tiedostot uudelleen (OK).*/

/*Tämä ohjelma ei vertaa (OK)-tiedostoja avattavien kanssa. Käyttäjän
pitää myöskin tarkistaa error-kansioon laitetut tiedostot itse (tekee
errorLoq:in). Tallentaa objekteja tietokantaan tiedostosta vaikka
error tapahtuisi keskellä tiedostoa.(Laittaa talteen errorLoq:iin
rivin missä virhe tapahtui.)*/

/*Käyttäjä voi päättää miltä riviltä lähtien ohjelma lähtee
tallentamaan tiedostosta objekteja tietokantaan (saveFromLine = rivi
errorLoqista) parametrin avulla (tiedoston manuaalisen korjauksen
jälkeen).*/
ClassMethod lueTiedosto(saveFromLine = 0)
{
           set directory = "C:\Users\TT\Desktop\Juho\App1Testi"

           set objectNumber = 0
           set objectLine = 0
           set error = 0
           set oRivi = 0
           set i = 0

           do showFiles(directory)
           if (error = 1) {write "Yhtään avattavaa HTML-tiedostoa ei
           löytynyt kansiosta: ", !, directory quit}

           set i = 1
           for {
           set cf = $$compareFiles()
           if (cf = 0){
           set file = ##class(%File).%New(fileNamePaths(i))
           do file.Open("R",2)
           if 'file.IsOpen {write !, "Tiedosto: ", !, fileNamePaths(i),
           !, "ei auennut!" quit}

           do readFile(file)
           if (error = 0){
           write !, objectNumber, " Objektia luotiin tietokantaan
           tiedostosta: ", !, fileNamePaths(i)
```

```
            set className = "write ^" _ $ClassName() _ "D"
            write !, "Objekteja tietokannassa yhteensä: " Xecute
            className
            set objectNumber = 0
            write !, viiva
            do file.Close()
            set newFileName = $$renameFileName(fileNamePaths(i))
            do renameFile(fileNamePaths(i), newFileName)
            }
            else {
            do file.Close()
            do moveFileToError(fileNamePaths(i), fileNames(i))
            do createErrorLog(fileNames(i))
            write !!, objectNumber, " Objektia luotiin tietokantaan tie-
            dostosta: ", !, fileNamePaths(i)
            write !, viiva
            set objectLine = 0
            set error = 0
            set objectNumber = 0
            do pressToContinue()
            write !, viiva
            }          }
            set i = i + 1
            quit:($get(fileNamePaths(i)) = "")
            }
            quit

showFiles(dir)
            set rs = ##class(%ResultSet).%New("%Library.File:FileSet")
            do rs.Execute(dir, "*", "Type")
            write "Avatut HTML-tiedostot: "

            while rs.Next(){
            set fileNamePath = rs.GetData(1)
            set fileName = rs.GetData(6) .
            if '(fileName?.e1"(OK)"1"."1"html"){
            if (fileNamePath?.e1"."1"html"){
            set i = $increment(i)
            write !, fileNamePath
            set fileNamePaths(i) = fileNamePath
            set fileNames(i) = fileName
                        }
            else{
            if (fileName?.e1".".e){
            do moveFileToError(fileNamePath, fileName)
            write !, "Tiedosto: '" , fileName, "' siirrettiin error
            kansioon!"
            do pressToContinue()
            }          }          }          }
            if $get(fileNamePaths(i)) = "" {
            do rs.Close()
            set error = 1
            quit
            }
            do viiva()
            write !, viiva
            do rs.Close()
            quit

moveFileToError(errorFile, newFileLocation)
```

```
            if $extract(directory, $length(directory)) '= "\"
            {set errorDir = directory _ "\Error"}
            else {set errorDir = directory _ "Error"}
            if '##class(%File).DirectoryExists(errorDir){do
            ##class(%File).CreateNewDir(directory, "Error")}
            set newErrorFile = errorDir _ "\" _ newFileLocation
            set fileToMove = ##class(%File).Rename(errorFile,
            newErrorFile)
            quit

viiva()
            set pituus = $length(fileNamePaths(i))
            set viiva = "-"
            for i = 1:1:pituus {set viiva = viiva _ "-"}
            if ($length(viiva) > 80){set viiva = $extract(viiva, 1, 80)}
            quit

compareFiles()
            set cf = 0
            for j = 1:1 {
            if $data(fileNamePaths(i+j)){
            set cf = ##class(%File).Compare(fileNamePaths(i),
            fileNamePaths(i+j))
            if (cf = 1){
            write !, "Tiedosto: ", !, fileNamePaths(i), !, "on
            identtinen tiedoston: ",!, fileNamePaths(i+j), " kanssa!"
            do moveFileToError(fileNamePaths(i), fileNames(i))
            write !, viiva
            do pressToContinue()
            write !, viiva
            quit
            }          }
            quit:($data(fileNamePaths(i+j)) = 0)
            }
            quit cf

pressToContinue()
            read !, ">>>Press any key to continue<<<", y#1:3
            quit

readFile(file)
            while 'file.AtEnd {
            set line = file.ReadLine()
            if (line?.E1"<TD><tt>".E1"</tt></TD>"){
            set objectLine = $increment(objectLine)
            set oLine(objectLine) = line
            set line = $$editLine(line)
            do createObject(line)
            }
            if (error = 1){quit}
            }
            if (objectLine > 0) && (objectLine < 6){
            set error = 1 set errorText = "Puuttuu rivejä!" write !,
            "ERROR: objectLine < 6; tiedostossa: ", !, fileNamePaths(i),
            !, "ei ole tarvittavia rivejä, jotka vaaditaan
            tallennettavaksi tietokantaan!"
            }
            quit

editLine(line)
```

```
            set line = $piece($piece(line,">",3),"<",1)
            if (line?1.2N1"."1.2N1"."2.4N1" "1.2N1":"1.2N1":"1.2N){
            set line = $replace(line, ".", "/")
            set line = $ZDATETIME($ZDATETIMEH(line, 4), 3)
            }
            quit line

createObject(line)
            if (objectLine = 1){set
            App1Oref=##class(ZenApp1.App1).%New()
            set App1Oref.DateAndTime=line}
            if (objectLine = 2){set App1Oref.EventType=line}
            if (objectLine = 3){set App1Oref.Computer=line}
            if (objectLine = 4){set App1Oref.SourceName=line}
            if (objectLine = 5){set App1Oref.EventId=line}
            if (objectLine = 6){set App1Oref.Message=line
            if (oRivi = saveFromLine){
            set Save=App1Oref.%Save()
            if '$$$ISERR(Save){
            set objectNumber = $increment(objectNumber)
            }
            else {
            do $System.Status.DisplayError(Save)
            set errorText = $System.Status.GetErrorText(Save)
            set error = 1
            }          }
            else {
            set oRivi = $increment(oRivi)
            }
            set counter = objectLine
            set objectLine = 0
            }
            quit

createErrorLog(fns)
            set fc = ##class(%Stream.FileCharacter).%New()
            set fc.Filename = errorDir _ "\errorLoq.txt"
            do fc.MoveToEnd()
            do fc.WriteLine("Aika: " _ $zdatetime($h, 3))
            do fc.WriteLine("File: " _ fns)
            do fc.WriteLine("Virhe: " _ errorText)
            do fc.WriteLine("Rivi: " _  (saveFromLine + objectNumber))
            if (objectNumber > 0){ for k = 1:1:counter {do
            fc.WriteLine(oLine(k))}}
            do fc.WriteLine(viiva)
            do fc.%Save()
            quit

renameFileName(fileName)
            set fileNamePiece1 = $piece(fileName, ".", 1)
            set fileNamePiece2 = $piece(fileName, ".", 2)
            set newFileName = fileNamePiece1 _ "(OK)" _ "." _
fileNamePiece2
            quit newFileName

renameFile(oFileName, newFileName)
            set renamedFile = ##class(%File).Rename(oFileName,
newFileName)
            if $$$ISERR(renamedFile){
            do $System.Status.DisplayError(renamedFile)
```

```
            write !, "Tiedostoa: ", !, oFileName, !, "ei voitu nimetä
            uudelleen tiedostoksi: "
            write !, newFileName, !, viiva
            do pressToContinue()
            }
            quit
}           }
```

Liite 2. Ensimmäinen sivu virhelokiraportista

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<TABLE BORDER="1">
<!--
<TR>
  <TH COLSPAN=4 BGCOLOR="BLACK"><FONT COLOR=WHITE>New W3SVC Messages
in System Event Log</FONT></TH>
</TR>
-->
<TR>
  <TH ALIGN=LEFT><tt>Time</tt></TH>
  <TH ALIGN=LEFT><tt>Event Type</tt></TH>
  <TH ALIGN=LEFT><tt>Computer</tt></TH>
  <TH ALIGN=LEFT><tt>Source Name</tt></TH>
  <TH ALIGN=LEFT><tt>Event Id</tt></TH>
  <TH ALIGN=LEFT><tt>Message</tt></TH>
</TR>

<TR bgCOLOR="#C0C0C0">
  <TD><tt>7.8.2013 07:31:04</tt></TD>
  <TD><tt>Warning event</tt></TD>
  <TD><tt>XXXXXXXX10.XXX.fi</tt></TD>
  <TD><tt>BizTalk Server</tt></TD>
  <TD><tt>5743</tt></TD>
  <TD><tt>The adapter failed to transmit message going to send port
"XX_XXXXXXXXX-XXX-XXX_OW_MLLP" with URL "XX.XXX.XXX.XXX:XXXXX". It
will be retransmitted after the retry interval specified for this Send
Port. Details:"MLLP Send adapter timed out while receiving the ACK.".
</tt></TD>
</TR>

<TR bgCOLOR="White">
  <TD><tt>7.8.2013 07:31:04</tt></TD>
  <TD><tt>Error event</tt></TD>
  <TD><tt>XXXXXXXX10.XXX.fi</tt></TD>
  <TD><tt>BizTalk Accelerator for HL7</tt></TD>
  <TD><tt>8451</tt></TD>
  <TD><tt>Unable to receive ACK from network due to error "MLLP Send
adapter timed out while receiving the ACK.". Username:
XXX\XXXXXXXX_XXXXXXX_X Time: 8/7/2013 7:31:04 AM </tt></TD>
</TR>

<TR bgCOLOR="#C0C0C0">
  <TD><tt>7.8.2013 07:30:09</tt></TD>
  <TD><tt>Warning event</tt></TD>
  <TD><tt>XXXXXXXX10.XXX.fi</tt></TD>
  <TD><tt>BizTalk Server</tt></TD>
  <TD><tt>5743</tt></TD>
  <TD><tt>The adapter failed to transmit message going to send port
"XX_XXXXXXXXXX-XXX-XXX_OW_MLLP" with URL "XX.XXX.XXX.XXX:XXXXX". It
will be retransmitted after the retry interval specified for this Send
Port. Details:"MLLP Send adapter timed out while receiving the ACK.".
</tt></TD>
</TR>
```

Liite 3. Caché Zen taulukko -luokka

```
/// Created using the page template: Default
Class ZenApp1.HomePage Extends %ZEN.Component.page
{

/// Class name of application this page belongs to.
Parameter APPLICATION = "ZenApp1.Application";

/// Displayed name of this page.
Parameter PAGENAME = "HomePage";

/// Domain used for localization.
Parameter DOMAIN;

/// This Style block contains page-specific CSS style definitions.
XData Style
{
<style type="text/css">
    /*Style classes for use on this page */

     body { font-family: arial; }

     table.tpTable caption{
       background: transparent;
       font-size: 1.4em;
       font-weight: bold;
       text-align: left;
       border: none;
     }

    /* even rows */
     .tpEven { color: black; background: #ebf3ff; }

    /* odd rows */
     .tpOdd { color: black; background: white; }

    /* this is a selected row */
     table.tpTable tr.tpSelected { background: #3d80df; color: yellow;
}

    /* hover for odd and even rows */
     tr.tpOdd:hover,tr.tpEven:hover {
        background-color: #3d80df;
        color: #ffffff; }

    /* table header style */
    table.tpTable th {
        border-right: 1px solid gray;
        border-bottom: 1px solid gray;
        background: #C5D6D6;
        color: black;
        font-weight: bold;
        text-align: left;
        padding: 2px;
        overflow: hidden;
     }

    /* selected column header (th) */
     table.tpTable th.tpHdrSelected { background: #3d80df; }
```

```
    /* filter layour */
     table.tpFilterLayout td { border: none; background: #C5D6D6; }
  </style>
}

/// This XML block defines the contents of this page.
XData Contents [ XMLNamespace = "http://www.intersystems.com/zen" ]
{
<page title="App1 Data" xmlns="http://www.intersystems.com/zen">

<dataController id="App1Data" modelClass="ZenApp1.App1" modelId=""/>

  <hgroup width="100%">

    <vgroup width="1%"></vgroup>

    <vgroup width="98%">

                      <tableNavigatorBar id="App1Nav"
tablePaneId="App1Table"/>

                      <tablePane
                      invalidMessage="Jokin meni pieleen!"
                      headerLayout="headersOnTop"
                      width=""
                      id="App1Table"
                      tableName="ZenApp1.App1"
                      maxRows=""
                      pageSize="100"
                      showRowNumbers="true"
                      showZebra="true"
                      useSnapshot="true"
                      extraColumnWidth="1%"
                      caption="App1 Data"
                      valueColumn="ID"
                      msgNoResult="Yhtään objektia ei löytynyt
tietokannasta!"
                      rowSelect="1"
                      enableToggleSelect="true">
                      <column colName="ID" hidden="true"/>
                      <column header="Date And Time"  width=""
colName="DateAndTime"           filterType="datetime"
filterOp="BETWEEN"/>
                      <column header="Event Type"     width=""
colName="EventType"   filterType="text" filterOp="UP["/>
                      <column header="Computer"       width=""
colName="Computer"    filterType="text" filterOp="UP["/>
                      <column header="Source Name"    width=""
colName="SourceName"  filterType="text" filterOp="UP["/>
                      <column header="Event Id"       width=""
colName="EventId"     filterType="text" filterOp="UP["/>
                      <column header="Message"        width=""
colName="Message"     filterType="text" filterOp="["/>
                      </tablePane>

           </vgroup>

    <vgroup width="1%"></vgroup></hgroup></page>}}
```

Liite 4. Virhetilanteen tapahtumalokitiedot Biztalkissa

```
Log Name:      Application
Source:        BizTalk Server
Date:          10/9/2014 6:17:16 PM
Event ID:      5743
Task Category: BizTalk Server
Level:         Warning
Keywords:      Classic
User:          N/A
Computer:      XXXXXXXX10.XXX.fi
Description:
The adapter failed to transmit message going to send port
"TO_XXXX_OW_HTTP" with URL
"https://XX.XXXXXXXXXXXXX.XX.com/XXXXXXX/XXXXXXXXXXX/XXXHTTPReceive.dl
l?XX". It will be retransmitted after the retry interval specified for
this Send Port. Details:"The underlying connection was closed: An un-
expected error occurred on a send.".
Event Xml:
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="BizTalk Server" />
    <EventID Qualifiers="32960">5743</EventID>
    <Level>3</Level>
    <Task>1</Task>
    <Keywords>0x80000000000000</Keywords>
    <TimeCreated SystemTime="2014-10-09T15:17:16.000000000Z" />
    <EventRecordID>94143330</EventRecordID>
    <Channel>Application</Channel>
    <Computer>XXXXXXXX10.XXX.fi</Computer>
    <Security />
  </System>
  <EventData>
    <Data>TO_HPSM_OW_HTTP</Data>
    <Da-
ta>https://XX.XXXXXXXXXXXXX.XX.com/XXXXXXX/XXXXXXXXXXX/XXXHTTPReceive.
dll?XX</Data>
    <Data>The underlying connection was closed: An unexpected error
occurred on a send.</Data>
  </EventData></Event>
```

Liite 5. Caché Zen Application -luokka

```
/// ZenApp1.Application
Class ZenApp1.Application Extends %ZEN.application
{

/// This is the name of this application.
Parameter APPLICATIONNAME = "ZenApp1";

/// This is the URL of the main starting page of this application.
Parameter HOMEPAGE = "ZenApp1.HomePage.cls";

/// This Style block contains application-wide CSS style definitions.
XData Style
{
<style type="text/css">
</style>
}

}
```